**Министр науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа № 5

Выполнил студент группы № M32071

Реброва Татьяна Ивановна

Проверил:

Бутенко Олег Романович

Санкт-Петербург  
2022

KafkaTutorialConsumerApplication.java  
  
package com.example;  
  
import com.example.entities.Color;  
import com.example.entities.Kotik;  
import com.example.entities.Owner;  
import com.example.services.MainService;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.Bean;  
  
import java.time.LocalDate;  
  
@SpringBootApplication  
public class KafkaTutorialConsumerApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(KafkaTutorialConsumerApplication.class, args);  
 }  
  
 @Bean  
 CommandLineRunner run(MainService mainService) {  
 return args -> {  
 Owner owner = new Owner("Nikita", LocalDate.of(2002, 03, 29), "user", "1234", "user");  
 mainService.saveOwner(owner);  
 Kotik kotik = new Kotik("Rubi", LocalDate.of(2022, 03, 29), "No name", Color.GREY, owner);  
 kotik.setOwnerId(owner);  
 mainService.saveKotik(kotik);  
 //mainService.addOwnerToKotik(owner.getId(), kotik.getId());  
  
 Owner owner2 = new Owner("Tanya", LocalDate.of(2002, 05, 29), "user2", "1234", "admin");  
 mainService.saveOwner(owner2);  
 Kotik kotik2 = new Kotik("Vudi", LocalDate.of(2022, 03, 29), "No name", Color.WHITE, owner);  
 kotik2.setOwnerId(owner2);  
 mainService.saveKotik(kotik2);  
 //mainService.addOwnerToKotik(owner2.getId(), kotik2.getId());  
  
 /\*Friend friend = mainService.addFriends(kotik.getId(), kotik2.getId());  
 mainService.saveFriend(friend);\*/  
 };  
 }  
}  
  
  
  
Color.java  
  
package com.example.entities;  
  
  
import java.util.HashMap;  
import java.util.Map;  
  
public enum Color {  
 BLACK("black"),  
 WHITE("white"),  
 RED("red"),  
 GREY("grey"),  
 GREEN("green"),  
 YELLOW("yellow");  
  
 private final String name;  
  
 Color(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 private static final Map<String, Color> LOOKUP\_MAP = new HashMap<>();  
  
 static {  
 for (Color env : values()) {  
 LOOKUP\_MAP.put(env.getName(), env);  
 }  
 }  
  
 public static Color getTypeByName(String url) {  
 return LOOKUP\_MAP.get(url);  
 }  
}  
  
  
Friend.java  
  
package com.example.entities;  
  
  
import com.example.wrapper.FriendWrap;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
  
@Entity  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Table (name = "kotiki\_friends")  
public class Friend{  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "kotik")  
 private Kotik kotik;  
 @OneToOne  
 @JoinColumn(name = "friend")  
 private Kotik friend;  
  
 public Friend(Kotik kotik, Kotik friend) {  
 this.kotik = kotik;  
 this.friend = friend;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public Kotik getKotik() {  
 return kotik;  
 }  
  
 public void setKotik(Kotik kotik) {  
 this.kotik = kotik;  
 }  
  
 public Kotik getFriend() {  
 return friend;  
 }  
  
 public void setFriend(Kotik friend) {  
 this.friend = friend;  
 }  
  
 public FriendWrap getFriendWrap(){  
 return new FriendWrap(id, kotik.getId(), friend.getId());  
 }  
}  
  
  
Kotik.java  
  
package com.example.entities;  
  
  
import com.example.tools.KotikiException;  
import com.example.wrapper.KotikWrap;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.List;  
  
@Entity  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Table (name = "kotiki")  
public class Kotik{  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
  
 private String name;  
 private LocalDate birthday;  
 private String breed;  
  
 @Column(name = "color")  
 private Color color;  
  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "owner")  
 private Owner owner;  
  
 @OneToMany(mappedBy = "kotik", orphanRemoval = true)  
 private final List<Friend> friends = new ArrayList<>();  
  
 public Kotik(String name, LocalDate birthday, String breed,  
 Color color, Owner owner) {  
 this.name = name;  
 this.birthday = birthday;  
 this.breed = breed;  
 this.color = color;  
 this.owner = owner;  
 }  
  
 public Friend addFriend(Kotik kotik) throws KotikiException {  
 if(kotik == null) {  
 throw new KotikiException("Sorry not found kotik");  
 }  
 if(kotik == this) {  
 throw new KotikiException("Sorry you can't be frendship with yourself I must to say your owner about your schiza");  
 }  
 Friend newFriend = new Friend(this, kotik);  
 friends.add(newFriend);  
 return newFriend;  
 }  
  
 public void removeFriends(Friend friend) {  
 friends.remove(friend);  
 }  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public LocalDate getBirthDay() {  
 return birthday;  
 }  
  
 public void setBirthDay(LocalDate birthDay) {  
 this.birthday = birthDay;  
 }  
  
 public String getBreed() {  
 return breed;  
 }  
  
 public void setBreed(String breed) {  
 this.breed = breed;  
 }  
  
 public Color getColorId() {  
 return color;  
 }  
  
 public void setColorId(Color color) {  
 this.color = color;  
 }  
  
 public Owner getOwnerId() {  
 return owner;  
 }  
  
 public void setOwnerId(Owner owner) {  
 this.owner = owner;  
 }  
  
 public KotikWrap getKotikWrap(){  
 return new KotikWrap(id, name, birthday, breed, color.ordinal(), owner.getId());  
 }  
  
}  
  
  
  
Owner.java  
  
package com.example.entities;  
  
  
import com.example.wrapper.OwnerWrap;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.List;  
  
@Entity  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Table(name = "owners")  
public class Owner{  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
 @Column(name = "name")  
 private String name;  
 private LocalDate birthday;  
 private String username;  
 private String password;  
 private String role;  
 @OneToMany(mappedBy = "owner", cascade = CascadeType.ALL, orphanRemoval = true)  
 private List<Kotik> kotiki;  
  
 public Owner(String name, LocalDate birthday, String username, String password, String role) {  
 this.name = name;  
 this.birthday = birthday;  
 this.username = username;  
 this.password = password;  
 this.role = role;  
 kotiki = new ArrayList<>();  
 }  
  
 public void addKotik(Kotik kotik) {  
 kotik.setOwnerId(this);  
 kotiki.add(kotik);  
 }  
  
 public void removeKotik(Kotik kotik) {  
 kotiki.remove(kotik);  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public LocalDate getBirthDay() {  
 return birthday;  
 }  
  
 public void setBirthDay(LocalDate birthDay) {  
 this.birthday = birthDay;  
 }  
  
 public List<Kotik> getKotiki() {  
 return kotiki;  
 }  
  
 public List<Integer> getKotikiId(){  
 List<Integer> kotikiId = new ArrayList<>();  
 for (Kotik item: kotiki) {  
 kotikiId.add(item.getId());  
 }  
 return kotikiId;  
 }  
 public LocalDate getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(LocalDate birthday) {  
 this.birthday = birthday;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getRole() {  
 return role;  
 }  
  
 public void setRole(String role) {  
 this.role = role;  
 }  
  
 public void setKotiki(List<Kotik> kotiki) {  
 this.kotiki = kotiki;  
 }  
  
  
 public OwnerWrap getOwnerWrap(){  
 return new OwnerWrap(id, name, birthday, username, password, role, this.getKotikiId());  
 }  
}  
  
  
  
FriendRepository.java  
  
package com.example.repository;  
  
import com.example.entities.Friend;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface FriendRepository extends JpaRepository<Friend, Integer> {  
}  
  
  
  
KotikRepository.java  
  
package com.example.repository;  
  
import com.example.entities.Color;  
import com.example.entities.Kotik;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import java.util.List;  
  
@Repository  
public interface KotikRepository extends JpaRepository<Kotik, Integer> {  
 List<Kotik> findByColor(Color color);  
}  
  
  
  
OwnerRepository.java  
  
package com.example.repository;  
  
import com.example.entities.Owner;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface OwnerRepository extends JpaRepository<Owner, Integer> {  
 Owner findByUsername(String username);  
}  
  
  
  
MainService.java  
  
package com.example.services;  
  
import com.example.entities.Color;  
import com.example.entities.Friend;  
import com.example.entities.Kotik;  
import com.example.entities.Owner;  
import com.example.wrapper.FriendWrap;  
import com.example.wrapper.KotikWrap;  
import com.example.wrapper.OwnerWrap;  
  
import java.time.LocalDate;  
import java.util.List;  
  
public interface MainService {  
 void createOwner(OwnerWrap ownerWrap);  
 Owner findOwnerById(int id);  
 Owner findOwnerByUsername(String username);  
 Owner saveOwner(Owner owner);  
// void deleteOwner(Owner owner);  
 void findAllOwners(List<OwnerWrap> ownerWraps);  
  
 void consProduceOwners(List<OwnerWrap> ownerWraps);  
 void consProduceOwner(OwnerWrap ownerWrap);  
 void consProduceKotiks(List<KotikWrap> kotikWraps);  
 void consProduceKotik(KotikWrap kotikWrap);  
 void consProduceFriends(List<FriendWrap> friendWraps);  
 void consProduceFriend(FriendWrap friendWrap);  
  
 Color getColor(int color) throws Exception;  
 void createKotik(KotikWrap kotikWrap) throws Exception;  
 Kotik findKotikById(int id);  
 Kotik saveKotik(Kotik kotik);  
// void deleteKotik(Kotik kotik);  
 void findAllKotiks(List<KotikWrap> kotikWraps);  
 List<Kotik> findKotikByColor(String color);  
//  
// Friend findFriendById(int id);  
 Friend saveFriend(Friend friend);  
// void deleteFriend(Friend friend);  
 void findAllFriends(List<FriendWrap> friendWraps);  
// void addOwnerToKotik(int idOwner, int idKotik);  
 void addFriends(FriendWrap friendWrap);  
 void getUserKotiks(String username);  
}  
  
  
MainServiceImpl.java  
  
package com.example.services;  
  
import com.example.entities.Color;  
import com.example.entities.Friend;  
import com.example.entities.Kotik;  
import com.example.entities.Owner;  
import com.example.repository.FriendRepository;  
import com.example.repository.KotikRepository;  
import com.example.repository.OwnerRepository;  
import com.example.wrapper.FriendWrap;  
import com.example.wrapper.KotikWrap;  
import com.example.wrapper.OwnerWrap;  
import lombok.RequiredArgsConstructor;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.kafka.annotation.KafkaListener;  
import org.springframework.kafka.core.KafkaTemplate;  
import org.springframework.stereotype.Service;  
  
import javax.transaction.Transactional;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@Service  
@RequiredArgsConstructor  
@Transactional  
@Slf4j  
public class MainServiceImpl implements MainService {  
 private final KotikRepository kotikRepo;  
 private final OwnerRepository ownerRepo;  
 private final FriendRepository friendRepo;  
  
 @Autowired  
 private KafkaTemplate<String, List> kafkaOwnersTemplate;  
  
 @Autowired  
 private KafkaTemplate<String, OwnerWrap> kafkaOwnerTemplate;  
  
 @Autowired  
 private KafkaTemplate<String, List> kafkaKotiksTemplate;  
  
 @Autowired  
 private KafkaTemplate<String, KotikWrap> kafkaKotikTemplate;  
  
 @Autowired  
 private KafkaTemplate<String, List> kafkaFriendsTemplate;  
  
 @Autowired  
 private KafkaTemplate<String, FriendWrap> kafkaFriendTemplate;  
  
 @Override  
 @KafkaListener(topics = "createOwner", groupId = "owner\_group\_id")  
 public void createOwner(OwnerWrap ownerWrap) {  
 log.info("Create owner");  
 Owner owner = new Owner(ownerWrap.getName(), ownerWrap.getBirthday(), ownerWrap.getUsername(),  
 ownerWrap.getPassword(), ownerWrap.getRole());  
 ownerRepo.save(owner);  
 consProduceOwner(owner.getOwnerWrap());  
 }  
  
 @Override  
 @KafkaListener(topics = "createKotik", groupId = "kotik\_group\_id")  
 public void createKotik(KotikWrap kotikWrap) throws Exception {  
 log.info("Create kotik");  
 Color colorEnum = getColor(kotikWrap.getColorId());  
 Owner owner = findOwnerById(kotikWrap.getOwner\_id());  
 Kotik kotik = new Kotik(kotikWrap.getName(), kotikWrap.getBirthday(), kotikWrap.getBreed(), colorEnum, owner);  
 ownerRepo.save(owner);  
 kotikRepo.save(kotik);  
 consProduceKotik(kotik.getKotikWrap());  
 }  
  
 @Override  
 public Owner findOwnerById(int id) {  
 log.info("Finding owner by id");  
 return ownerRepo.getById(id);  
 }  
  
 @Override  
 @KafkaListener(topics = "findOwnerByUsername", groupId = "owner\_group\_id")  
 public Owner findOwnerByUsername(String username) {  
 log.info("Finding owner by username");  
 Owner owner = ownerRepo.findByUsername(username);  
 OwnerWrap ownerWrap = owner.getOwnerWrap();  
 System.out.println("BLAAAAAAAAAAAAAAAAAAAAAAAAAAAAAT");  
 consProduceOwner(ownerWrap);  
 return owner;  
 }  
  
 @Override  
 public Owner saveOwner(Owner owner) {  
 log.info("Saving new owner");  
 return ownerRepo.save(owner);  
 }  
//  
// @Override  
// public void deleteOwner(Owner owner) {  
// log.info("Delete owner");  
// ownerRepo.delete(owner);  
// }  
  
 @Override  
 @KafkaListener(topics = "owners", groupId = "owners\_group\_id")  
 public void findAllOwners(List<OwnerWrap> ownerWraps) {  
 log.info("Fetching all owners");  
 ownerWraps = ownerRepo.findAll().stream().  
 map((owner -> owner.getOwnerWrap())).  
 collect(Collectors.toList());  
 consProduceOwners(ownerWraps);  
 }  
  
 @Override  
 public void consProduceOwners(List<OwnerWrap> ownerWraps) {  
 kafkaOwnersTemplate.send("sendOwners", ownerWraps);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void consProduceOwner(OwnerWrap ownerWrap) {  
 kafkaOwnerTemplate.send("sendOwner", ownerWrap);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void consProduceKotiks(List<KotikWrap> kotikWraps) {  
 kafkaOwnersTemplate.send("sendKotiks", kotikWraps);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void consProduceKotik(KotikWrap kotikWrap) {  
 kafkaKotikTemplate.send("sendKotik", kotikWrap);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void consProduceFriends(List<FriendWrap> friendWraps) {  
 kafkaFriendsTemplate.send("sendFriends", friendWraps);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void consProduceFriend(FriendWrap friendWrap) {  
 kafkaFriendTemplate.send("sendFriend", friendWrap);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public Color getColor(int color) throws Exception{  
 for (Color colors: Color.values()) {  
 if (colors.ordinal() == color) {  
 return colors;  
 }  
 }  
 throw new Exception("Error");  
 }  
  
 @Override  
 public Kotik findKotikById(int id) {  
 log.info("Finding kotik by id");  
 return kotikRepo.getById(id);  
 }  
  
 @Override  
 public Kotik saveKotik(Kotik kotik) {  
 log.info("Saving new kotik");  
 return kotikRepo.save(kotik);  
 }  
//  
// @Override  
// public void deleteKotik(Kotik kotik) {  
// log.info("Delete kotik");  
// kotikRepo.delete(kotik);  
// }  
//  
 @Override  
 @KafkaListener(topics = "kotiks", groupId = "kotiks\_group\_id")  
 public void findAllKotiks(List<KotikWrap> kotikWraps) {  
 log.info("Fetching all kotiks");  
 kotikWraps = kotikRepo.findAll().stream().  
 map((kotik -> kotik.getKotikWrap())).  
 collect(Collectors.toList());  
 consProduceKotiks(kotikWraps);  
 }  
  
 @Override  
 @KafkaListener(topics = "findKotikByColor", groupId = "kotiks\_group\_id")  
 public List<Kotik> findKotikByColor(String color) {  
 log.info("Find kotik by color");  
 Color colorEnum = Color.getTypeByName(color);  
 List<Kotik> kotiki = kotikRepo.findByColor(colorEnum);  
 consProduceKotiks(kotiki.stream().map(kotik -> kotik.getKotikWrap()).collect(Collectors.toList()));  
 return kotiki;  
 }  
  
// @Override  
// public Friend findFriendById(int id) {  
// log.info("Finding friend by id");  
// return friendRepo.getById(id);  
// }  
//  
 @Override  
 public Friend saveFriend(Friend friend) {  
 log.info("Saving new friend");  
 return friendRepo.save(friend);  
 }  
//  
// @Override  
// public void deleteFriend(Friend friend) {  
// log.info("Delete friend");  
// friendRepo.delete(friend);  
// }  
//  
 @Override  
 @KafkaListener(topics = "friends", groupId = "friends\_group\_id")  
 public void findAllFriends(List<FriendWrap> friendWraps) {  
 log.info("Fetching all friends");  
 friendWraps = friendRepo.findAll().stream().  
 map((friend -> friend.getFriendWrap())).  
 collect(Collectors.toList());  
 consProduceFriends(friendWraps);  
 }  
//  
// @Override  
// public void addOwnerToKotik(int idOwner, int idKotik) {  
// log.info("Adding owner to kotik");  
// Owner owner = findOwnerById(idOwner);  
// Kotik kotik = findKotikById(idKotik);  
// owner.addKotik(kotik);  
// kotik.setOwnerId(owner);  
// ownerRepo.save(owner);  
// kotikRepo.save(kotik);  
// }  
//  
 @Override  
 @KafkaListener(topics = "createFriend", groupId = "friend\_group\_id")  
 public void addFriends(FriendWrap friendWrap) {  
 Kotik kotik = findKotikById(friendWrap.kotikId);  
 Kotik kotikFriend = findKotikById(friendWrap.friendId);  
 Friend newFriend = kotik.addFriend(kotikFriend);  
 friendRepo.save(newFriend);  
 consProduceFriend(newFriend.getFriendWrap());  
 }  
  
 @Override  
 @KafkaListener(topics = "userKotiks", groupId = "kotiks\_group\_id")  
 public void getUserKotiks(String username) {  
 Owner owner = findOwnerByUsername(username);  
 List<Kotik> kotiks = kotikRepo.findAll();  
 List<Kotik> result = new ArrayList<>();  
 for (Kotik kotik:kotiks) {  
 if(kotik.getOwner().getId() == owner.getId()) {  
 result.add(kotik);  
 }  
 }  
 consProduceKotiks(result.stream().map(kotik -> kotik.getKotikWrap()).collect(Collectors.toList()));  
 }  
}  
  
  
  
KotikiException.java  
  
package com.example.tools;  
  
public class KotikiException extends RuntimeException {  
 public KotikiException() {  
 }  
  
 public KotikiException(String message) {  
 super(message);  
 }  
}  
  
  
  
FriendWrap.java  
  
package com.example.wrapper;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
  
@Data  
public class FriendWrap {  
 public int id;  
 public int kotikId;  
 public int friendId;  
  
 public FriendWrap() {}  
  
 public FriendWrap(int id, int kotikId, int friendId) {  
 this.id = id;  
 this.kotikId = kotikId;  
 this.friendId = friendId;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public int getKotikId() {  
 return kotikId;  
 }  
  
 public void setKotikId(int kotikId) {  
 this.kotikId = kotikId;  
 }  
  
 public int getFriendId() {  
 return friendId;  
 }  
  
 public void setFriendId(int friendId) {  
 this.friendId = friendId;  
 }  
  
 @Override  
 public String toString() {  
 return "FriendWrap{" +  
 "id=" + id +  
 ", kotikId=" + kotikId +  
 ", friendId=" + friendId +  
 '}';  
 }  
}  
  
  
  
KotikWrap.java  
  
package com.example.wrapper;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import com.example.entities.Color;  
import java.time.LocalDate;  
  
@Data  
public class KotikWrap {  
 private int id;  
 private String name;  
 private LocalDate birthday;  
 private String breed;  
 private int colorId;  
 private int owner\_id;  
  
 public KotikWrap() {}  
  
 public KotikWrap(int id, String name, LocalDate birthday, String breed, int colorId, int owner\_id) {  
 this.id = id;  
 this.name = name;  
 this.birthday = birthday;  
 this.breed = breed;  
 this.colorId = colorId;  
 this.owner\_id = owner\_id;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public LocalDate getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(LocalDate birthday) {  
 this.birthday = birthday;  
 }  
  
 public String getBreed() {  
 return breed;  
 }  
  
 public void setBreed(String breed) {  
 this.breed = breed;  
 }  
  
 public int getColorId() {  
 return colorId;  
 }  
  
 public void setColorId(int colorId) {  
 this.colorId = colorId;  
 }  
  
 public int getOwner\_id() {  
 return owner\_id;  
 }  
  
 public void setOwner\_id(int owner\_id) {  
 this.owner\_id = owner\_id;  
 }  
  
 @Override  
 public String toString() {  
 return "KotikWrap{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", birthday=" + birthday +  
 ", breed='" + breed + '\'' +  
 ", colorId=" + colorId +  
 ", owner\_id=" + owner\_id +  
 '}';  
 }  
}  
  
  
  
OwnerWrap.java  
  
package com.example.wrapper;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import com.example.entities.Kotik;  
  
import java.time.LocalDate;  
import java.util.List;  
  
@Data  
//@AllArgsConstructor  
public class OwnerWrap {  
 private int id;  
 private String name;  
 private LocalDate birthday;  
 private String username;  
 private String password;  
 private String role;  
 private List<Integer> kotikiId;  
  
  
 @Override  
 public String toString() {  
 return "OwnerWrap{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", birthday=" + birthday +  
 ", username='" + username + '\'' +  
 ", password='" + password + '\'' +  
 ", role='" + role + '\'' +  
 ", kotikiId=" + kotikiId +  
 '}';  
 }  
  
 public OwnerWrap(int id, String name, LocalDate birthday, String username, String password, String role, List<Integer> kotikiId) {  
 this.id = id;  
 this.name = name;  
 this.birthday = birthday;  
 this.username = username;  
 this.password = password;  
 this.role = role;  
 this.kotikiId = kotikiId;  
 }  
  
 public OwnerWrap() {}  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public LocalDate getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(LocalDate birthday) {  
 this.birthday = birthday;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getRole() {  
 return role;  
 }  
  
 public void setRole(String role) {  
 this.role = role;  
 }  
  
 public List<Integer> getKotikiId() {  
 return kotikiId;  
 }  
  
 public void setKotikiId(List<Integer> kotikiId) {  
 this.kotikiId = kotikiId;  
 }  
}  
  
  
  
KafkaTutorialConsumerApplicationTests.java  
  
package com.example;  
  
import org.junit.jupiter.api.Test;  
import org.springframework.boot.test.context.SpringBootTest;  
  
@SpringBootTest  
class KafkaTutorialConsumerApplicationTests {  
  
 @Test  
 void contextLoads() {  
 }  
  
}  
  
  
  
KafkaTutorialProducerApplication.java  
  
package com.example;  
  
import com.fasterxml.jackson.databind.ObjectMapper;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;  
import org.springframework.context.annotation.Bean;  
  
@SpringBootApplication(exclude={DataSourceAutoConfiguration.class})  
public class KafkaTutorialProducerApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(KafkaTutorialProducerApplication.class, args);  
 }  
  
}  
  
  
  
MainController.java  
  
package com.example.controller;  
  
import com.example.model.Message;  
import com.example.producer.ProducerService;  
import com.example.services.MainService;  
import com.example.wrapper.FriendWrap;  
import com.example.wrapper.KotikWrap;  
import com.example.wrapper.OwnerWrap;  
import lombok.RequiredArgsConstructor;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.kafka.core.KafkaTemplate;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.web.bind.annotation.\*;  
//import org.springframework.web.servlet.support.ServletUriComponentsBuilder;  
//import org.springframework.security.core.userdetails.User;  
//import org.springframework.security.core.userdetails.UserDetails;  
//import org.springframework.security.core.userdetails.UserDetailsService;  
//import org.springframework.security.core.userdetails.UsernameNotFoundException;  
  
import java.net.URI;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@RestController  
@RequiredArgsConstructor  
public class MainController {  
 @Autowired  
 private MainService mainService;  
 @Autowired  
 private KafkaTemplate<String, List> kafkaOwnersTemplate;  
 @Autowired  
 private KafkaTemplate<String, String> kafkaOwnerTemplate;  
 @Autowired  
 private KafkaTemplate<String, OwnerWrap> kafkaOwnerWrapTemplate;  
 @Autowired  
 private KafkaTemplate<String, List> kafkaKotiksTemplate;  
 @Autowired  
 private KafkaTemplate<String, String> kafkaKotikTemplate;  
 @Autowired  
 private KafkaTemplate<String, KotikWrap> kafkaKotikWrapTemplate;  
 @Autowired  
 private KafkaTemplate<String, List> kafkaFriendsTemplate;  
 @Autowired  
 private KafkaTemplate<String, String> kafkaFriendTemplate;  
 @Autowired  
 private KafkaTemplate<String, FriendWrap> kafkaFriendWrapTemplate;  
  
 @GetMapping("/error")  
 public String error() {  
 return "Error";  
 }  
  
 @GetMapping("/")  
 public String hello() {  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 return "Hello "+auth.getName();  
 }  
  
 @GetMapping("/user")  
 public String user() {  
 return "User";  
 }  
  
 @GetMapping("/admin")  
 public String admin() {  
 return "Admin";  
 }  
  
 @GetMapping("/admin/owner/owners")  
 public ResponseEntity<List<OwnerWrap>> owners() throws InterruptedException {  
 List<OwnerWrap> owners = new ArrayList<>();  
 kafkaOwnersTemplate.send("owners", owners);  
 Thread.sleep(1000);  
 List<OwnerWrap> ownerWraps = mainService.ownersWrap;  
 return ResponseEntity.ok().body(ownerWraps);  
 }  
  
 @GetMapping("/admin/owner/findByUsername")  
 public ResponseEntity<OwnerWrap> findByUsername(@RequestParam String username) {  
 kafkaOwnerTemplate.send("findOwnerByUsername", username);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 OwnerWrap ownerWrap = mainService.ownerWrap;  
 return ResponseEntity.ok().body(ownerWrap);  
 }  
  
 @GetMapping("/admin/kotik/kotiks")  
 public ResponseEntity<List<KotikWrap>>getKotiks() throws InterruptedException {  
 List<KotikWrap> kotiks = new ArrayList<>();  
 kafkaKotiksTemplate.send("kotiks", kotiks);  
 Thread.sleep(1000);  
 List<KotikWrap> kotikWraps = mainService.kotiksWrap;  
 return ResponseEntity.ok().body(kotikWraps);  
 }  
  
 @GetMapping("/admin/friend/friends")  
 public ResponseEntity<List<FriendWrap>>getFriends() throws InterruptedException {  
 List<FriendWrap> friends = new ArrayList<>();  
 kafkaFriendsTemplate.send("friends", friends);  
 Thread.sleep(1000);  
 List<FriendWrap> friendWraps = mainService.friendsWrap;  
 return ResponseEntity.ok().body(friendWraps);  
 }  
  
 @PostMapping("admin/owner/createOwner")  
 public ResponseEntity<OwnerWrap> createOwner(@RequestParam String name, @RequestParam String date, @RequestParam String username, @RequestParam String password, @RequestParam String role) {  
 List<Integer> kotiks = new ArrayList<>();  
 OwnerWrap ownerWrap = new OwnerWrap(0, name, LocalDate.parse(date), username, password, role, kotiks);  
 kafkaOwnerWrapTemplate.send("createOwner", ownerWrap);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 OwnerWrap ownerWrap1 = mainService.ownerWrap;  
 return ResponseEntity.ok().body(ownerWrap1);  
 }  
  
 @PostMapping("/admin/kotik/createkotik")  
 public ResponseEntity<KotikWrap> createKotik(@RequestParam String name, @RequestParam String date, @RequestParam String breed, @RequestParam int color, @RequestParam int ownerId) throws Exception {  
 KotikWrap kotikWrap = new KotikWrap(0, name, LocalDate.parse(date), breed, color, ownerId);  
 kafkaKotikWrapTemplate.send("createKotik", kotikWrap);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 KotikWrap kotikWrap1 = mainService.kotikWrap;  
 return ResponseEntity.ok().body(kotikWrap1);  
 }  
  
 @PostMapping("/admin/friend/createFriend")  
 public ResponseEntity<FriendWrap>addFriend(@RequestParam int idKotik, @RequestParam int idFriend) {  
 FriendWrap friendWrap = new FriendWrap(0,idKotik, idFriend);  
 kafkaFriendWrapTemplate.send("createFriend", friendWrap);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 FriendWrap friendWrap1 = mainService.friendWrap;  
 return ResponseEntity.ok().body(friendWrap1);  
 }  
  
 @GetMapping("/user/kotik/kotiks")  
 public ResponseEntity<List<KotikWrap>>getUserKotiks(){  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 kafkaOwnerTemplate.send("userKotiks", auth.getName());  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 List<KotikWrap> kotikWraps = mainService.kotiksWrap;  
 return ResponseEntity.ok().body(kotikWraps);  
 };  
  
 @GetMapping("/admin/kotik/findByColor")  
 public ResponseEntity<List<KotikWrap>>findKotikByColor(@RequestParam String color) {  
 kafkaKotikTemplate.send("findKotikByColor", color);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 List<KotikWrap> kotikWraps = mainService.kotiksWrap;  
 return ResponseEntity.ok().body(kotikWraps);  
 }  
  
 @PostMapping("/user/kotik/findByColor")  
 public ResponseEntity<List<KotikWrap>>getUserKotiksByColor(@RequestParam String color){  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 List<KotikWrap> result = new ArrayList<>();  
 kafkaOwnerTemplate.send("findOwnerByUsername", auth.getName());  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 OwnerWrap ownerWrap = mainService.ownerWrap;  
 kafkaKotikTemplate.send("findKotikByColor", color);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 List<KotikWrap> kotikWraps1 = mainService.kotiksWrap;  
 List<KotikWrap> kotikWraps2 = new ArrayList<>();  
 kotikWraps2 = kotikWraps1;  
 for(KotikWrap kotikWrap:kotikWraps2) {  
 if(kotikWrap.getOwner\_id() == ownerWrap.getId()) {  
 result.add(kotikWrap);  
 }  
 }  
 return ResponseEntity.ok().body(result);  
 }  
}  
  
  
  
ProducerService.java  
  
package com.example.producer;  
  
import com.example.model.Message;  
import com.example.wrapper.OwnerWrap;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.kafka.annotation.KafkaListener;  
import org.springframework.kafka.core.KafkaTemplate;  
import org.springframework.stereotype.Service;  
  
import javax.print.attribute.standard.Media;  
  
@Service  
public class ProducerService {  
 @Autowired  
 private KafkaTemplate<String, Message> kafkaTemplate;  
  
 public void produce(Message message) {  
 System.out.println("Producing the message: " + message);  
 kafkaTemplate.send("messages", message);  
 }  
  
 @KafkaListener(topics = "messagesFinal", groupId = "message\_group\_id")  
 public void produceConsumer(Message message) {  
 System.out.println("Consumer th message: " + message);  
 }  
  
}  
  
  
  
CustomAuthencationProvider.java  
  
package com.example.security;  
  
import com.example.services.MainService;  
import com.example.tools.KotikiException;  
import com.example.wrapper.OwnerWrap;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.kafka.core.KafkaTemplate;  
import org.springframework.security.authentication.AuthenticationProvider;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.stereotype.Component;  
  
@Component  
public class CustomAuthencationProvider implements AuthenticationProvider {  
 @Autowired  
 private KafkaTemplate<String, String> kafkaTemplate;  
  
 @Autowired  
 private MainService mainService;  
  
 @Override  
 public Authentication authenticate(Authentication authentication) throws AuthenticationException {  
 String username = authentication.getName();  
 String password = authentication.getCredentials().toString();  
  
 kafkaTemplate.send("findOwnerByUsername", username);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 OwnerWrap owner = mainService.ownerWrap;  
 if(owner == null) {  
 throw new KotikiException("Unknown owner" + username);  
 }  
 if(!password.equals(owner.getPassword())) {  
 throw new KotikiException("Bad password");  
 }  
 UserDetails principal = User.builder()  
 .username(owner.getUsername())  
 .password(owner.getPassword())  
 .roles(owner.getRole())  
 .build();  
 return new UsernamePasswordAuthenticationToken(principal, password, principal.getAuthorities());  
  
 }  
  
 @Override  
 public boolean supports(Class<?> authentication) {  
 return authentication.equals(UsernamePasswordAuthenticationToken.class);  
 }  
}  
  
  
  
SecurityConfig.java  
  
package com.example.security;  
  
  
import com.example.services.CustomUserDetailsService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.crypto.password.NoOpPasswordEncoder;  
  
@EnableWebSecurity(debug = true)  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
 @Autowired  
 private CustomUserDetailsService userDetailsService;  
  
 @Autowired  
 private CustomAuthencationProvider customAuthencationProvider;  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return NoOpPasswordEncoder.getInstance();  
 }  
  
 @Override  
 public void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.userDetailsService(userDetailsService);  
 }  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.authorizeRequests()  
 .antMatchers("/user/\*\*").hasAnyRole("user", "admin")  
 .antMatchers("/admin/\*\*").hasRole("admin")  
 .antMatchers("/").permitAll()  
 .and()  
 .formLogin();  
 http.csrf().disable();  
 }  
  
  
}  
  
  
  
CustomUserDetailsService.java  
  
package com.example.services;  
  
import com.example.wrapper.OwnerWrap;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.kafka.annotation.KafkaListener;  
import org.springframework.kafka.core.KafkaTemplate;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.stereotype.Service;  
  
@Service  
public class CustomUserDetailsService implements UserDetailsService {  
 @Autowired  
 private KafkaTemplate<String, String> kafkaTemplate;  
  
 @Autowired  
 private MainService mainService;  
  
 @Override  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
 kafkaTemplate.send("findOwnerByUsername", username);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 OwnerWrap owner = mainService.ownerWrap;  
 if (owner == null) {  
 throw new UsernameNotFoundException("Unknown user: "+ username);  
 }  
 UserDetails user = User.builder()  
 .username(owner.getUsername())  
 .password(owner.getPassword())  
 .roles(owner.getRole())  
 .build();  
 return user;  
 }  
}  
  
  
MainService.java  
  
package com.example.services;  
  
import com.example.model.Message;  
import com.example.wrapper.FriendWrap;  
import com.example.wrapper.KotikWrap;  
import com.example.wrapper.OwnerWrap;  
import com.fasterxml.jackson.core.JsonProcessingException;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.kafka.annotation.KafkaListener;  
import org.springframework.kafka.core.KafkaTemplate;  
import org.springframework.stereotype.Service;  
  
import javax.print.attribute.standard.Media;  
import java.util.List;  
  
@Service  
@Slf4j  
public class MainService {  
 @Autowired  
 private KafkaTemplate<String, Message> kafkaTemplate;  
  
 public List<OwnerWrap> ownersWrap;  
 public OwnerWrap ownerWrap;  
 public List<KotikWrap> kotiksWrap;  
 public KotikWrap kotikWrap;  
 public List<FriendWrap> friendsWrap;  
 public FriendWrap friendWrap;  
  
 @KafkaListener(topics = "sendOwners", groupId = "owners\_group\_id")  
 public List getUsers(List<OwnerWrap> ownersWrap) {  
 this.ownersWrap = ownersWrap;  
 System.out.println(ownersWrap);  
 return this.ownersWrap;  
 }  
  
 @KafkaListener(topics = "sendOwner", groupId = "owner\_group\_id")  
 public OwnerWrap getOwner(OwnerWrap ownerWrap) {  
 this.ownerWrap = ownerWrap;  
 System.out.println(ownerWrap);  
 return this.ownerWrap;  
 }  
  
 @KafkaListener(topics = "sendKotiks", groupId = "kotiks\_group\_id")  
 public List getKotiks(List<KotikWrap> kotiksWrap) {  
 this.kotiksWrap = kotiksWrap;  
 System.out.println(kotiksWrap);  
 return this.kotiksWrap;  
 }  
  
 @KafkaListener(topics = "sendKotik", groupId = "kotik\_group\_id")  
 public KotikWrap getKotik(KotikWrap kotikWrap) {  
 this.kotikWrap = kotikWrap;  
 System.out.println(kotikWrap);  
 return this.kotikWrap;  
 }  
  
 @KafkaListener(topics = "sendFriends", groupId = "friends\_group\_id")  
 public List getFriends(List<FriendWrap> friendsWrap) {  
 this.friendsWrap = friendsWrap;  
 System.out.println(friendsWrap);  
 return this.kotiksWrap;  
 }  
  
 @KafkaListener(topics = "sendFriend", groupId = "friend\_group\_id")  
 public FriendWrap getFriend(FriendWrap friendWrap) {  
 this.friendWrap = friendWrap;  
 System.out.println(ownerWrap);  
 return this.friendWrap;  
 }  
}  
  
  
  
KotikiException.java  
  
package com.example.tools;  
  
public class KotikiException extends RuntimeException {  
 public KotikiException() {  
 }  
  
 public KotikiException(String message) {  
 super(message);  
 }  
}  
  
  
  
FriendWrap.java  
  
package com.example.wrapper;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
  
@Data  
public class FriendWrap {  
 public int id;  
 public int kotikId;  
 public int friendId;  
  
 public FriendWrap() {}  
  
 public FriendWrap(int id, int kotikId, int friendId) {  
 this.id = id;  
 this.kotikId = kotikId;  
 this.friendId = friendId;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public int getKotikId() {  
 return kotikId;  
 }  
  
 public void setKotikId(int kotikId) {  
 this.kotikId = kotikId;  
 }  
  
 public int getFriendId() {  
 return friendId;  
 }  
  
 public void setFriendId(int friendId) {  
 this.friendId = friendId;  
 }  
  
 @Override  
 public String toString() {  
 return "FriendWrap{" +  
 "id=" + id +  
 ", kotikId=" + kotikId +  
 ", friendId=" + friendId +  
 '}';  
 }  
}  
  
  
  
KotikWrap.java  
  
package com.example.wrapper;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import java.time.LocalDate;  
  
@Data  
public class KotikWrap {  
 private int id;  
 private String name;  
 private LocalDate birthday;  
 private String breed;  
 private int colorId;  
 private int owner\_id;  
  
 public KotikWrap() {}  
  
 public KotikWrap(int id, String name, LocalDate birthday, String breed, int colorId, int owner\_id) {  
 this.id = id;  
 this.name = name;  
 this.birthday = birthday;  
 this.breed = breed;  
 this.colorId = colorId;  
 this.owner\_id = owner\_id;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public LocalDate getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(LocalDate birthday) {  
 this.birthday = birthday;  
 }  
  
 public String getBreed() {  
 return breed;  
 }  
  
 public void setBreed(String breed) {  
 this.breed = breed;  
 }  
  
 public int getColorId() {  
 return colorId;  
 }  
  
 public void setColorId(int colorId) {  
 this.colorId = colorId;  
 }  
  
 public int getOwner\_id() {  
 return owner\_id;  
 }  
  
 public void setOwner\_id(int owner\_id) {  
 this.owner\_id = owner\_id;  
 }  
  
 @Override  
 public String toString() {  
 return "KotikWrap{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", birthday=" + birthday +  
 ", breed='" + breed + '\'' +  
 ", colorId=" + colorId +  
 ", owner\_id=" + owner\_id +  
 '}';  
 }  
}  
  
  
  
OwnerWrap.java  
  
package com.example.wrapper;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
  
import java.time.LocalDate;  
import java.util.List;  
  
//@Data  
public class OwnerWrap {  
 private int id;  
 private String name;  
 private LocalDate birthday;  
 private String username;  
 private String password;  
 private String role;  
 private List<Integer> kotikiId;  
  
 @Override  
 public String toString() {  
 return "OwnerWrap{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", birthday=" + birthday +  
 ", username='" + username + '\'' +  
 ", password='" + password + '\'' +  
 ", role='" + role + '\'' +  
 ", kotikiId=" + kotikiId +  
 '}';  
 }  
  
 public OwnerWrap(int id, String name, LocalDate birthday, String username, String password, String role, List<Integer> kotikiId) {  
 this.id = id;  
 this.name = name;  
 this.birthday = birthday;  
 this.username = username;  
 this.password = password;  
 this.role = role;  
 this.kotikiId = kotikiId;  
 }  
  
 public OwnerWrap() {}  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public LocalDate getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(LocalDate birthday) {  
 this.birthday = birthday;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getRole() {  
 return role;  
 }  
  
 public void setRole(String role) {  
 this.role = role;  
 }  
  
 public List<Integer> getKotikiId() {  
 return kotikiId;  
 }  
  
 public void setKotikiId(List<Integer> kotikiId) {  
 this.kotikiId = kotikiId;  
 }  
}  
  
  
  
KafkaTutorialProducerApplicationTests.java  
  
package com.example;  
  
import org.junit.jupiter.api.Test;  
import org.springframework.boot.test.context.SpringBootTest;  
  
@SpringBootTest  
class KafkaTutorialProducerApplicationTests {  
  
 @Test  
 void contextLoads() {  
 }  
  
}