МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ №14 - 16

по дисциплине

«Шаблоны программных платформ языка Java

Вариант 21

Выполнил студент группы ИКБО-20-19 Николаев-Аксенов И. С.

Принял Батанов А. О.
Ассистент

Практические работы выполнены «___»_____2021 г. (подпись студента)

«Зачтено» «___»_____2021 г. (подпись руководителя)

Москва 2021

Содержание

Практическая работа №14

Цель работы

Тема: Знакомство со Spring MVC. Работа с Rest API в Spring.

Постановка задачи: Создать отдельный репозиторий Git. Создать простой html-документ, который будет содержать вашу фамилию, имя, номер группы, номер варианта. Создать контроллер, который будет возвращать данный статический документ при переходе на url «/home». Выполнить задание в зависимости с вариантом индивидуального задания Создать класс Post с полями text, creationDate. Создать класс User с полями firstName, lastName, middleName, birthDate. Создать классы-контроллеры для создания, удаления объектов и получения всех объектов каждого типа. Сами объекты хранить в памяти.

Листинг программы

Application.java

```java
package PR14.Application;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

Post.java

```java
package PR14.Application.model;

import com.fasterxml.jackson.annotation.JsonProperty;

import java.util.Date;

public class Post {
    private final String text;
    private final Date creationDate;

    public Post(@JsonProperty("text") String text) {
        this.text = text;
        this.creationDate = new Date();
    }

    public String getText() {
        return text;
    }

    public Date getCreationDate() {
        return creationDate;
    }

    @Override
    public String toString() {
        return "Пост от " + creationDate + "\nТекст: " + text;
    }
}
```

User.java

```java
package PR14.Application.model;

import com.fasterxml.jackson.annotation.JsonProperty;

import java.util.ArrayList;
import java.util.List;

public class User {
    private final String firstName;
    private final String lastName;
    private final String middleName;
    private final String birthDate;

    private final List<Post> posts = new ArrayList<>();

    public User(@JsonProperty("firstName") String firstName,
                @JsonProperty("lastName") String lastName,
                @JsonProperty("middleName") String middleName,
                @JsonProperty("birthDate") String birthDate) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    public List<Post> getPosts() {
        return posts;
    }

    public void addPost(Post post) {
        this.posts.add(post);
    }

    public void deletePost(Post post) {
        this.posts.remove(post);
    }

    @Override
    public String toString() {
        return "Пользователь " + lastName + " " + firstName + " " + middleName + ", день
рождения: " + birthDate + "\nОпубликовал следующие посты: " + posts;
    }
}
```

UserPostHolder.java

```
package PR14.Application.model;

import com.fasterxml.jackson.annotation.JsonProperty;

public class UserPostHolder {
    private final User user;
    private final String text;

    public UserPostHolder(@JsonProperty("user") User user,
                          @JsonProperty("text") String text) {
        this.user = user;
        this.text = text;
    }

    public User getUser() {
        return user;
    }

    public String getText() {
        return text;
    }
}
```

UserDataAccesService.java

```
package PR14.Application.service;

import PR14.Application.model.Post;
import PR14.Application.model.User;
import PR14.Application.model.UserPostHolder;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;
import java.util.List;

@Repository
public class UserDataAccessService {
    private static List<User> DB = new ArrayList<>();

    public int insertUser(User user) {
        DB.add(user);
        return 1;
    }

    public int insertPost(UserPostHolder userPostHolder) {
        User user = userPostHolder.getUser();
        String text = userPostHolder.getText();
        for(User i : DB) {
            if(i.getFirstName().equals(user.getFirstName()) &&
i.getLastName().equals(user.getLastName()) &&
i.getMiddleName().equals(user.getMiddleName()) &&
i.getBirthDate().equals(user.getBirthDate())) {
                i.addPost(new Post(text));
            }
        }
        return 1;
    }
```

```
    public int deleteUser(User user) {
        DB.removeIf(i -> i.getFirstName().equals(user.getFirstName()) &&
i.getLastName().equals(user.getLastName()) &&
i.getMiddleName().equals(user.getMiddleName()) &&
i.getBirthDate().equals(user.getBirthDate()));
        return 1;
    }

    public int deletePost(UserPostHolder userPostHolder) {
        User user = userPostHolder.getUser();
        String text = userPostHolder.getText();
        for(User i : DB) {
            if(i.getFirstName().equals(user.getFirstName()) &&
i.getLastName().equals(user.getLastName()) &&
i.getMiddleName().equals(user.getMiddleName()) &&
i.getBirthDate().equals(user.getBirthDate())) {
                for(Post j : i.getPosts()) {
                    if(j.getText().equals(text)) {
                        i.deletePost(j);
                    }
                }
            }
        }
        return 1;
    }

    public List<User> getAllUsers() {
        return DB;
    }
}
```

UserService.java

```
package PR14.Application.service;

import PR14.Application.model.User;
import PR14.Application.model.UserPostHolder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserService {
    private final UserDataAccessService userDataAccessService;

    @Autowired
    public UserService(UserDataAccessService userDataAccessService) {
        this.userDataAccessService = userDataAccessService;
    }

    public int insertUser(User user) {
        return userDataAccessService.insertUser(user);
    }

    public int insertPost(UserPostHolder userPostHolder) {
        return userDataAccessService.insertPost(userPostHolder);
    }

    public int deleteUser(User user) {
```

```
        return userDataAccessService.deleteUser(user);
    }

    public int deletePost(UserPostHolder userPostHolder) {
        return userDataAccessService.deletePost(userPostHolder);
    }

    public List<User> getAllUsers() {
        return userDataAccessService.getAllUsers();
    }
}
```

HomeController.java

```
package PR14.Application.controller;

import org.springframework.http.MediaType;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class HomeController {
    @GetMapping(value = "/home", produces = MediaType.TEXT_HTML_VALUE)
    @ResponseBody
    public String homePage() {
        return "<html>\n" +
                "<head><title>Home</title></head>\n" +
                "<body>\n" +
                "Фамилия: Николаев-Аксенов<br><hr>\nИмя: Иван<br><hr>\nНомер группы:
ИКБО-20-19<br><hr>\nНомер варианта: 21(6)<hr>" +
                "</body>\n" +
                "</html>";
    }
}
```

UserController.java

```java
package PR14.Application.controller;

import PR14.Application.model.User;
import PR14.Application.model.UserPostHolder;
import PR14.Application.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class UserController {
    private final UserService userService;

    @Autowired
    public UserController(UserService userService) {
        this.userService = userService;
    }

    @PostMapping("/users")
    public int insertUser(@RequestBody User user) {
        return userService.insertUser(user);
    }

    @PostMapping("/posts")
    public int insertPost(@RequestBody UserPostHolder userPostHolder) {
        return userService.insertPost(userPostHolder);
    }

    @DeleteMapping("/users")
    public int deleteUser(@RequestBody User user) {
        return userService.deleteUser(user);
    }

    @DeleteMapping("/posts")
    public int deletePost(@RequestBody UserPostHolder userPostHolder) {
        return userService.deletePost(userPostHolder);
    }

    @GetMapping("/users")
    public List<User> getAllUsers() {
        return userService.getAllUsers();
    }
}
```

Результат выполнения программы

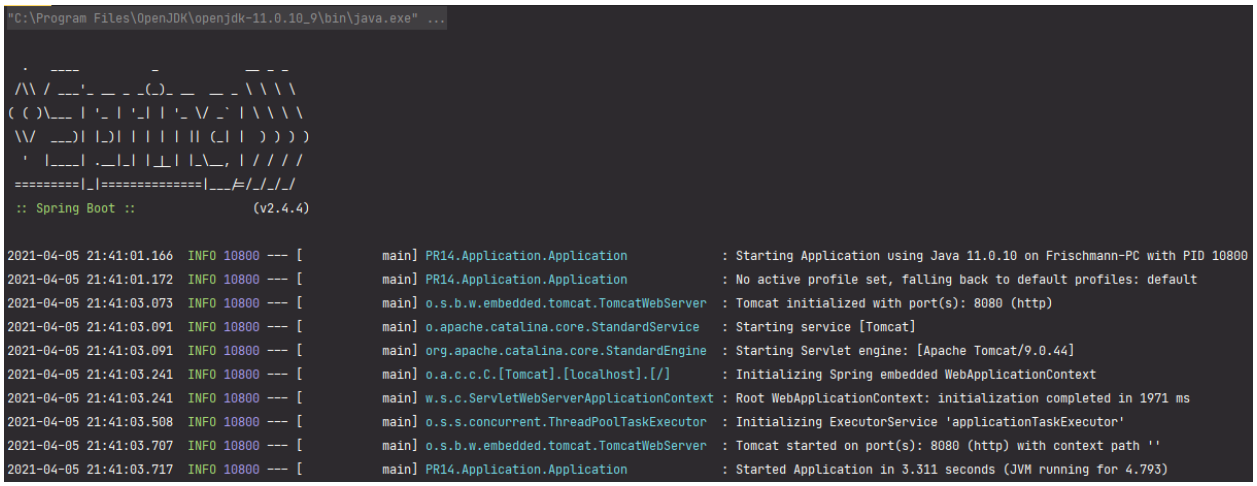


Рисунок 14.1 – Демонстрация работы программы

Рисунок 14.2 – Демонстрация работы программы

Рисунок 14.3 – Демонстрация работы программы

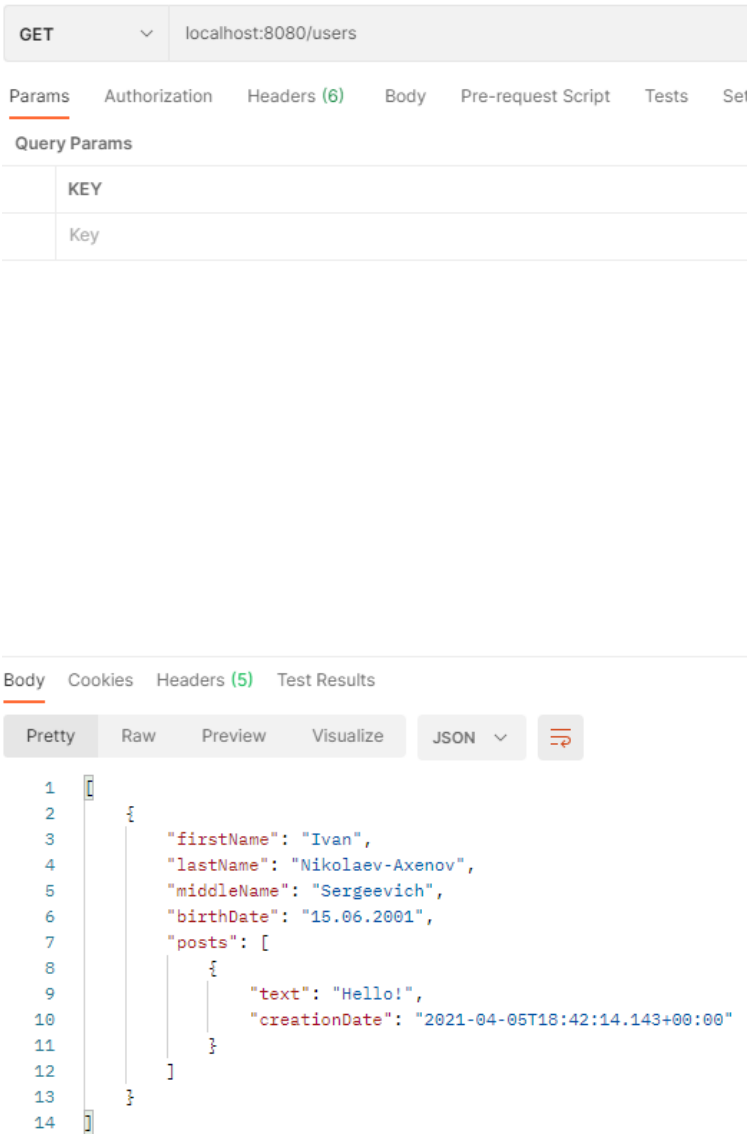


Рисунок 14.4 – Демонстрация работы программы

Практическая работа №15

Цель работы

Тема: Использование Hibernate в Spring framework.

Постановка задачи: Изменить программу с предыдущего задания так, чтобы объекты хранились в базе данных PostgreSQL вместо памяти компьютера.

Листинг программы

Application.java

```
package PR15.Application;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

User.java

```
package PR15.Application.model;

import com.sun.istack.NotNull;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;
import java.util.UUID;

@Entity
@Table(name = "users")
public class User implements Serializable {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "last_name")
    @NotNull
    private String lastName;

    @Column(name = "first_name")
    @NotNull
    private String firstName;

    @Column(name = "middle_name")
```

```
    @NotNull
    private String middleName;

    @Column(name = "birth_date")
    @NotNull
    private String birthDate;

    public User() {

    }

    public User(String lastName, String firstName, String middleName, String birthDate)
{
        this.lastName = lastName;
        this.firstName = firstName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public UUID getId() {
        return id;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    @Override
    public String toString() {
        return "Пользователь #" + id + " " + lastName + " " + firstName + " " +
middleName + ", день рождения: " + birthDate;
    }
}
```

Post.java

```java
package PR15.Application.model;

import com.sun.istack.NotNull;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.GenericGenerator;
import org.springframework.format.annotation.DateTimeFormat;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.Date;
import java.util.UUID;

@Entity
@Table(name = "posts")
public class Post {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "text")
    @NotNull
    private String text;

    @CreationTimestamp
    @Column(name = "creation_date")
    private LocalDateTime creationDate;

    @Column(name = "owner")
    @NotNull
    private UUID owner;

    public Post() {

    }

    public Post(String text, UUID owner) {
        this.text = text;
        this.owner = owner;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }

    public UUID getOwner() {
        return owner;
    }
}
```

UserService.java

```
package PR15.Application.service;

import PR15.Application.model.User;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.PostMapping;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.List;
import java.util.UUID;

@Service
public class UserService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;

    public UserService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
    }

    @PreDestroy
    public void unSession() {
        session.close();
    }

    public void addUser(User user) {
        session.beginTransaction();
        session.saveOrUpdate(user);
        session.getTransaction().commit();
    }

    public List<User> getUsers() {
        return session.createQuery("select u from User u", User.class).list();
    }

    public User getUser(UUID id) {
        return session.createQuery("select p from User u where u.id = p.id = '" + id +
"'", User.class).getSingleResult();
    }

    public void deleteUser(UUID id) {
        session.beginTransaction();

        User t = session.load(User.class, id);
        session.delete(t);

        session.getTransaction().commit();
    }
}
```

PostService.java

```
package PR15.Application.service;

import PR15.Application.model.Post;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.List;
import java.util.UUID;

@Service
public class PostService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;

    public PostService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
    }

    @PreDestroy
    public void unSession() {
        session.close();
    }

    public void addPost(Post post) {
        session.beginTransaction();
        session.saveOrUpdate(post);
        session.getTransaction().commit();
    }

    public List<Post> getPosts() {
        return session.createQuery("select p from Post p", Post.class).list();
    }

    public List<Post> getPost(UUID id) {
        return session.createQuery("select p from Post p where p.id ='" + id + "'",
Post.class).list();
    }

    public void deletePosts(Post post) {
        session.beginTransaction();

        List<Post> query = session.createQuery("select p from Post p where p.id = '" +
post.getId() + "'", Post.class).list();
        for (Post p : query) {
            session.delete(p);
        }

        session.getTransaction().commit();
    }
```

```
    public void deletePost(UUID id) {
        session.beginTransaction();

        Post t = session.load(Post.class, id);
        session.delete(t);

        session.getTransaction().commit();
    }
}
```

UserController.java

```
package PR15.Application.controller;

import PR15.Application.model.User;
import PR15.Application.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class UserController {
    @Autowired
    private UserService userService;

    @PostMapping("/users")
    public void addUser(@RequestBody User user) {
        userService.addUser(user);
    }

    @GetMapping("/users")
    public List<User> getUsers() {
        return userService.getUsers();
    }

    @GetMapping("/users/{id}")
    public User getUser(@PathVariable UUID id) {
        return userService.getUser(id);
    }

    @DeleteMapping("/users/{id}")
    public void deleteUser(@PathVariable UUID id) {
        userService.deleteUser(id);
    }
}
```

PostController.java

```java
package PR15.Application.controller;

import PR15.Application.model.Post;
import PR15.Application.service.PostService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class PostController {
    @Autowired
    private PostService postService;

    @PostMapping("/post")
    public void addPost(@RequestBody Post post) {
        postService.addPost(post);
    }

    @GetMapping("/posts")
    public List<Post> getAll() {
        return postService.getPosts();
    }

    @GetMapping("/post/{id}")
    public List<Post> getPost(@PathVariable UUID id) {
        return postService.getPost(id);
    }

    @DeleteMapping("/post/{id}")
    public void deletePost(@PathVariable UUID id) {
        postService.deletePost(id);
    }
}
```

Config.java

```
package PR15.Application.config;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.PlatformTransactionManager;

import javax.sql.DataSource;
import java.util.Properties;

@Configuration
public class Config {
    @Bean
    public HikariDataSource dataSource(){
        HikariConfig config = new HikariConfig();
        config.setJdbcUrl("jdbc:postgresql://localhost:5432/pr15db");
        config.setUsername("postgres");
        config.setPassword("secret");
        config.setDriverClassName("org.postgresql.Driver");
        return new HikariDataSource(config);
    }

    @Bean
    public LocalSessionFactoryBean sessionFactory(DataSource dataSource){
        LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();
        factoryBean.setDataSource(dataSource);
        factoryBean.setPackagesToScan("PR15.Application");
        Properties properties = new Properties();
        properties.setProperty("hibernate.dialect",
"org.hibernate.dialect.PostgreSQLDialect");
        factoryBean.setHibernateProperties(properties);
        return factoryBean;
    }

    @Bean
    public PlatformTransactionManager platformTransactionManager(LocalSessionFactoryBean
factoryBean){
        HibernateTransactionManager transactionManager = new
HibernateTransactionManager();
        transactionManager.setSessionFactory(factoryBean.getObject());
        return transactionManager;
    }
}
```

Результат выполнения программы

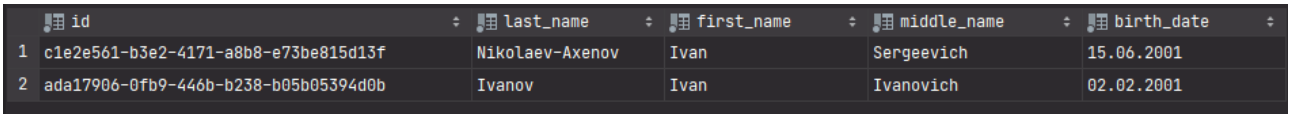
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.4)

2021-04-05 21:43:33.895  INFO 16920 --- [           main] PR15.Application.Application             : Starting Application using Java 11.0.10 on Frischmann-PC with PID 16920
2021-04-05 21:43:33.907  INFO 16920 --- [           main] PR15.Application.Application             : No active profile set, falling back to default profiles: default
2021-04-05 21:43:35.522  INFO 16920 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2021-04-05 21:43:35.532  INFO 16920 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-04-05 21:43:35.532  INFO 16920 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.44]
2021-04-05 21:43:35.657  INFO 16920 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-04-05 21:43:35.657  INFO 16920 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1697 ms
2021-04-05 21:43:35.719  INFO 16920 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2021-04-05 21:43:35.919  INFO 16920 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2021-04-05 21:43:36.067  INFO 16920 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.28.Final
2021-04-05 21:43:36.283  INFO 16920 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-04-05 21:43:36.412  INFO 16920 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect
2021-04-05 21:43:37.237  INFO 16920 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.trans
2021-04-05 21:43:37.355  WARN 16920 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queri
2021-04-05 21:43:37.429  INFO 16920 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2021-04-05 21:43:37.617  INFO 16920 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2021-04-05 21:43:37.627  INFO 16920 --- [           main] PR15.Application.Application             : Started Application in 4.154 seconds (JVM running for 5.126)
```

Рисунок 15.1 – Демонстрация работы программы

Рисунок 15.2 – Демонстрация работы программы

Рисунок 15.3 – Демонстрация работы программы



	id	last_name	first_name	middle_name	birth_date
1	c1e2e561-b3e2-4171-a8b8-e73be815d13f	Nikolaev-Axenov	Ivan	Sergeevich	15.06.2001
2	ada17906-0fb9-446b-b238-b05b05394d0b	Ivanov	Ivan	Ivanovich	02.02.2001

Рисунок 15.4 – Демонстрация работы программы

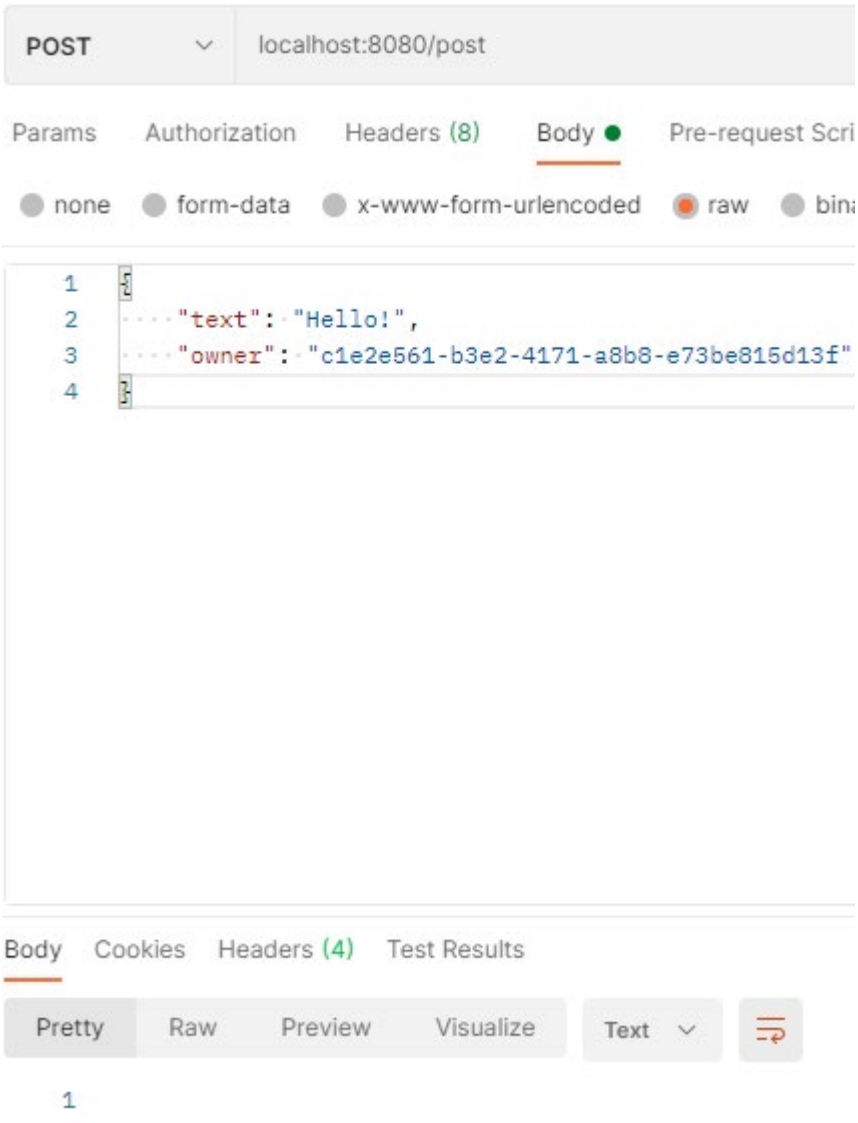


Рисунок 15.5 – Демонстрация работы программы

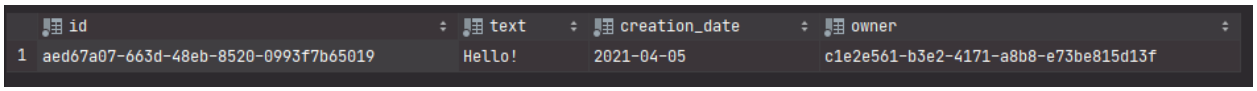


Рисунок 15.6 – Демонстрация работы программы

Практическая работа №16

Цель работы

Тема: Изучение видов связей между сущностями в Hibernate. Использование транзакций.

Постановка задачи: Создать связь Один-ко-многим между сущностями из предыдущего задания и проверить работу lazy loading.

Листинг программы

Application.java

```
package PR16.Application;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

User.java

```
package PR16.Application.model;

import com.sun.istack.NotNull;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;
import java.util.*;

@Entity
@Table(name = "users")
public class User implements Serializable {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "last_name")
    @NotNull
    private String lastName;

    @Column(name = "first_name")
    @NotNull
    private String firstName;
```

```
    @Column(name = "middle_name")
    @NotNull
    private String middleName;

    @Column(name = "birth_date")
    @NotNull
    private String birthDate;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Post> posts = new ArrayList<>();

    public User() {

    }

    public User(String lastName, String firstName, String middleName, String birthDate)
{
        this.lastName = lastName;
        this.firstName = firstName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public void addPost(Post post) {
        posts.add(post);
        post.setUser(this);
    }

    public void removePost(Post post) {
        posts.remove(post);
        post.setUser(null);
    }

    public UUID getId() {
        return id;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    @Override
    public String toString() {
        return "Пользователь #" + id + " " + lastName + " " + firstName + " " +
middleName + ", день рождения: " + birthDate;
    }
}
```

Post.java

```java
package PR16.Application.model;

import com.sun.istack.NotNull;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.GenericGenerator;
import org.springframework.format.annotation.DateTimeFormat;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.Date;
import java.util.UUID;

@Entity
@Table(name = "posts")
public class Post {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "text")
    @NotNull
    private String text;

    @CreationTimestamp
    @Column(name = "creation_date")
    private LocalDateTime creationDate;

    @ManyToOne(fetch = FetchType.LAZY)
    private User user;

    public Post() {

    }

    public Post(String text) {
        this.text = text;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }
}
```

UserService.java

```java
package PR16.Application.service;

import PR16.Application.model.Post;
import PR16.Application.model.User;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.List;
import java.util.UUID;

@Service
public class UserService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;

    public UserService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
    }

    @PreDestroy
    public void unSession() {
        session.close();
    }

    public void addUser(User user) {
        session.beginTransaction();
        session.saveOrUpdate(user);
        session.getTransaction().commit();
    }

    public void addPost(UUID id, Post post) {
        session.beginTransaction();

        User t = session.load(User.class, id);
        t.addPost(post);
        session.saveOrUpdate(t);

        session.getTransaction().commit();
    }

    public void removePost(UUID id, Post post) {
        session.beginTransaction();

        User t = session.load(User.class, id);
        t.removePost(post);
        session.saveOrUpdate(t);

        session.getTransaction().commit();
    }
```

```
    public List<User> getUsers() {
        return session.createQuery("select u from User u", User.class).list();
    }

    public User getUser(UUID id) {
        return session.createQuery("select u from User u where u.id = p.id = '" + id +
"'", User.class).getSingleResult();
    }

    public void deleteUser(UUID id) {
        session.beginTransaction();

        User t = session.load(User.class, id);
        session.delete(t);

        session.getTransaction().commit();
    }
}
```

PostService.java

```
package PR16.Application.service;

import PR16.Application.model.Post;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.List;
import java.util.UUID;

@Service
public class PostService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;

    public PostService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
    }

    @PreDestroy
    public void unSession() {
        session.close();
    }

    public void addPost(Post post) {
        session.beginTransaction();
        session.saveOrUpdate(post);
        session.getTransaction().commit();
```

```
    }

    public List<Post> getPosts() {
        return session.createQuery("select p from Post p", Post.class).list();
    }

    public List<Post> getPost(UUID id) {
        return session.createQuery("select p from Post p where p.id ='" + id + "'",
Post.class).list();
    }

    public void deletePosts(Post post) {
        session.beginTransaction();

        List<Post> query = session.createQuery("select p from Post p where p.id = '" +
post.getId() + "'", Post.class).list();
        for (Post p : query) {
            session.delete(p);
        }

        session.getTransaction().commit();
    }

    public void deletePost(UUID id) {
        session.beginTransaction();

        Post t = session.load(Post.class, id);
        session.delete(t);

        session.getTransaction().commit();
    }
}
```

UserController.java

```
package PR16.Application.controller;

import PR16.Application.model.Post;
import PR16.Application.model.User;
import PR16.Application.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class UserController {
    @Autowired
    private UserService userService;

    @PostMapping("/users")
    public void addUser(@RequestBody User user) {
        userService.addUser(user);
    }

    @GetMapping("/users")
    public List<User> getUsers() {
        return userService.getUsers();
    }
```

```
    @PostMapping("/userpost/{id}")
    public void addPost(@PathVariable UUID id, @RequestBody String text) {
        userService.addPost(id, new Post(text));
    }

    @DeleteMapping("/userpost/{id}")
    public void deletePost(@PathVariable UUID id, Post post) {
        userService.removePost(id, post);
    }

    @GetMapping("/users/{id}")
    public User getUser(@PathVariable UUID id) {
        return userService.getUser(id);
    }

    @DeleteMapping("/users/{id}")
    public void deleteUser(@PathVariable UUID id) {
        userService.deleteUser(id);
    }
}
```

PostController.java

```
package PR16.Application.controller;

import PR16.Application.model.Post;
import PR16.Application.service.PostService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class PostController {
    @Autowired
    private PostService postService;

    @PostMapping("/post")
    public void addPost(@RequestBody Post post) {
        postService.addPost(post);
    }

    @GetMapping("/posts")
    public List<Post> getPosts() {
        return postService.getPosts();
    }

    @GetMapping("/post/{id}")
    public List<Post> getPost(@PathVariable UUID id) {
        return postService.getPost(id);
    }

    @DeleteMapping("/post/{id}")
    public void deletePost(@PathVariable UUID id) {
        postService.deletePost(id);
    }
}
```

Config.java

```
package PR16.Application.config;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.PlatformTransactionManager;

import javax.sql.DataSource;
import java.util.Properties;

@Configuration
public class Config {
    @Bean
    public HikariDataSource dataSource(){
        HikariConfig config = new HikariConfig();
        config.setJdbcUrl("jdbc:postgresql://localhost:5432/pr16db");
        config.setUsername("postgres");
        config.setPassword("secret");
        config.setDriverClassName("org.postgresql.Driver");
        return new HikariDataSource(config);
    }

    @Bean
    public LocalSessionFactoryBean sessionFactory(DataSource dataSource){
        LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();
        factoryBean.setDataSource(dataSource);
        factoryBean.setPackagesToScan("PR16.Application");
        Properties properties = new Properties();
        properties.setProperty("hibernate.dialect",
"org.hibernate.dialect.PostgreSQLDialect");
        factoryBean.setHibernateProperties(properties);
        return factoryBean;
    }

    @Bean
    public PlatformTransactionManager platformTransactionManager(LocalSessionFactoryBean
factoryBean){
        HibernateTransactionManager transactionManager = new
HibernateTransactionManager();
        transactionManager.setSessionFactory(factoryBean.getObject());
        return transactionManager;
    }
}
```

Результат выполнения программы

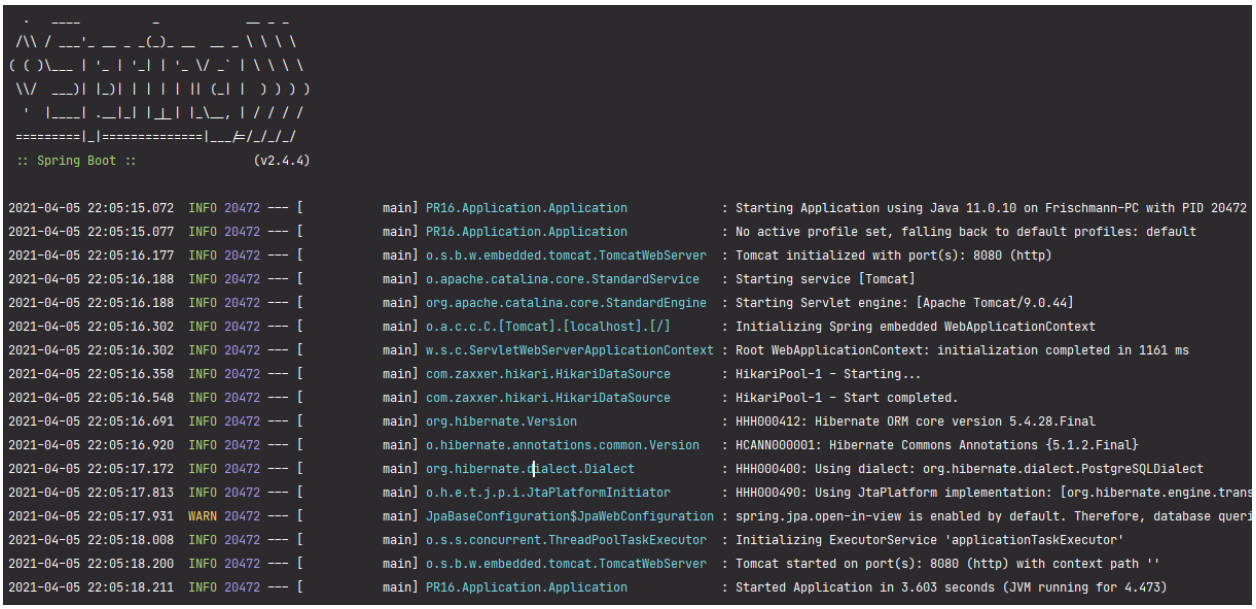


Рисунок 15.1 – Демонстрация работы программы

Рисунок 15.2 – Демонстрация работы программы

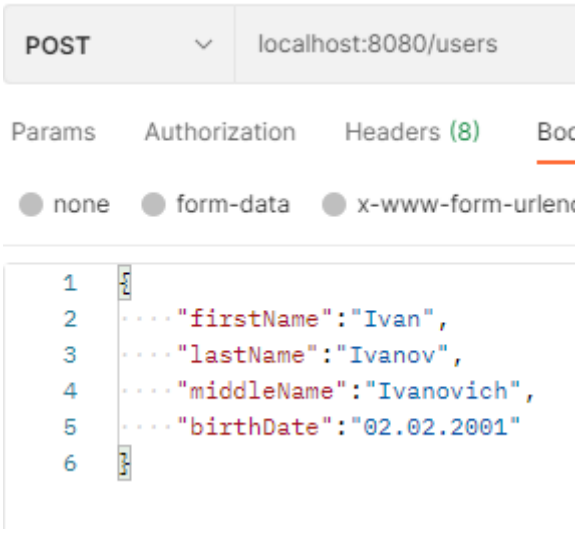


Рисунок 15.3 – Демонстрация работы программы

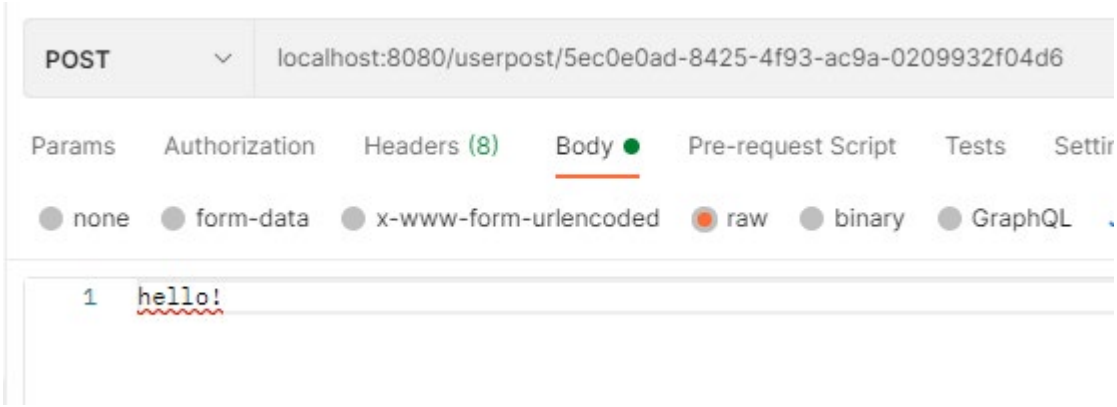


Рисунок 15.4 – Демонстрация работы программы

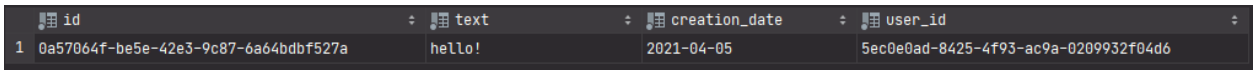


Рисунок 15.5 – Демонстрация работы программы

Вывод

В ходе выполнения данных практических работ были получены навыки работы с основными технологиями, необходимыми для создания клиент-серверных приложений. Также были получены навыки работы с фреймворком Spring.

Список использованных источников

1. Стелтинг С., Маасен О. Применение шаблонов Java. Библиотека профессионала.: Пер. с англ. — М.: Издательский дом "Вильяме", 2002. — 576 с.: ил. — Парал. тит. англ.
2. Functional Interfaces in Java: Fundamentals and Examples 1st ed. Edition, Kindle Edition [Электронный ресурс]. URL: https://www.amazon.com/Functional-Interfaces-Java-Fundamentals-Examples-ebook/dp/B07NRHQSCW (дата обращения: 29.01.21). Заголовок с экрана.
3. Hibernate Search 6.0.0.Final: Reference Documentation [Электронный ресурс]. URL: https://docs.jboss.org/hibernate/stable/search/reference/en-US/html_single/ (дата обращения: 29.01.21). Заголовок с экрана.
4. Паттерны проектирования на Java. Каталог Java-примеров. [Электронный ресурс]. URL: https://refactoring.guru/ru/design-patterns/java (дата обращения: 29.01.21). Заголовок с экрана.
5. Руководство по Spring [Электронный ресурс]. URL: https://proselyte.net/tutorials/spring-tutorial-full-version/ (дата обращения: 29.01.21). Заголовок с экрана.
6. The Reactive Manifesto [Электронный ресурс]. URL: https://www.reactivemanifesto.org/ (дата обращения: 29.01.21). Заголовок с экрана.
7. Spring Framework Documentation [Электронный ресурс]. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/web.html (дата обращения: 29.01.21). Заголовок с экрана.
8. Hibernate Search 6.0.0. Final: Reference Documentation [Электронный ресурс]. URL: https://docs.jboss.org/hibernate/stable/search/reference/en-US/html_single/ (дата обращения: 29.01.21). Заголовок с экрана.