Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра экономической информатики

Лабораторная работа №9

**Технология разработки приложений Spring REST**

Выполнила: Забелло К.В. Проверил:

Студентка: гр. 714301 Кунцевич А.А.

Минск 2020

**Цель –** разработать api-приложение для организации веб-сервиса на базе технологии Spring REST.

**Задание:** разработать приложение на базе технологии Spring REST для управления базой данных посредствам RestFul API для проведения CRUD операций с таблицой.

**Текст программы:**

package com.zabello.enums;

public enum City {

BREST,

MINSK,

MOGILEV,

VITEBSK,

HOMEL,

HRODNO

}

package com.zabello.model;

import lombok.Data;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.MappedSuperclass;

@Data

@MappedSuperclass

public class BaseEntity {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private long id;

}

package com.zabello.model;

import com.zabello.enums.City;

import lombok.Data;

import javax.persistence.\*;

import java.time.LocalDate;

@Entity

@Data

@Table(name = "users")

public class User extends BaseEntity{

@Column(name = "first\_name")

private String firstName;

@Column(name = "middle\_name")

private String middleName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "date\_of\_birth")

private LocalDate dateOfBirth;

@Column(name = "passport\_series")

private String passportSeries;

@Column(name = "passport\_number")

private String passportNumber;

@Column(name = "city")

@Enumerated(EnumType.STRING)

private City city;

@Column(name = "address")

private String address;

@Column(name = "phone\_number")

private String phoneNumber;

@Column(name = "email")

private String email;

@Column(name = "employment")

private Boolean employment;

@Column(name = "job\_position")

private String jobPosition;

@Column(name = "place\_of\_residence")

private String placeOfResidence;

@Column(name = "citizenship")

private String citizenship;

}

package com.zabello.repository;

import com.zabello.model.User;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface UserRepository extends JpaRepository<User, Long> {

}

package com.zabello.rest;

import com.zabello.model.User;

import com.zabello.service.UserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import javax.validation.Valid;

import java.util.Optional;

@RestController

@RequestMapping("/rest/user")

public class UserRestController {

@Autowired

private UserService service;

@GetMapping(value = "/{id}", produces = MediaType.APPLICATION\_XML\_VALUE)

public ResponseEntity<User> getUserById(@PathVariable Long id) {

if (id == null)

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

Optional<User> user = service.getByUserId(id);

if (!user.isPresent())

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

return new ResponseEntity<>(user.get(), HttpStatus.OK);

}

@PostMapping(produces = MediaType.APPLICATION\_XML\_VALUE)

public ResponseEntity<User> addUser(@RequestBody @Valid User user) {

HttpHeaders headers = new HttpHeaders();

if (user == null)

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

service.save(user);

return new ResponseEntity<>(user, headers, HttpStatus.CREATED);

}

@PutMapping(produces = MediaType.APPLICATION\_XML\_VALUE)

public ResponseEntity<User> updateUser(@RequestBody @Valid User user) {

if (user == null)

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

service.save(user);

return new ResponseEntity<>(user, HttpStatus.OK);

}

@DeleteMapping(value = "/{id}", produces = MediaType.APPLICATION\_XML\_VALUE)

public ResponseEntity<User> deleteUser(@PathVariable Long id) {

if (id == null)

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

service.delete(id);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

}

package com.zabello.rest;

import com.zabello.model.User;

import com.zabello.service.UserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import javax.validation.Valid;

import java.util.List;

@RestController

@RequestMapping("/rest/users")

public class UsersRestController {

@Autowired

private UserService service;

@GetMapping(produces = MediaType.APPLICATION\_XML\_VALUE)

public ResponseEntity<List<User>> getUsers(){

List<User> users = service.getAll();

if (users.isEmpty())

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

return new ResponseEntity<>(users, HttpStatus.OK);

}

@PutMapping(produces = MediaType.APPLICATION\_XML\_VALUE)

public ResponseEntity<List<User>> updateUsers(@RequestBody List<User> users){

if(users.isEmpty()){

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

}

service.updateAll(users);

return new ResponseEntity<>(users, HttpStatus.OK);

}

@DeleteMapping(produces = MediaType.APPLICATION\_XML\_VALUE)

public ResponseEntity<List<User>> deleteUsers(@RequestBody List<User> users){

if(users.isEmpty()){

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

}

service.deleteAll(users);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

}

package com.zabello.service;

import com.zabello.model.User;

import java.util.List;

import java.util.Optional;

public interface UserService {

Optional<User> getByUserId(Long id);

void save(User user);

void delete(Long id);

List<User> getAll();

void updateAll(List<User> users);

void deleteAll(List<User> users);

}

package com.zabello.service;

import com.zabello.model.User;

import java.util.List;

import java.util.Optional;

public interface UserService {

Optional<User> getByUserId(Long id);

void save(User user);

void delete(Long id);

List<User> getAll();

void updateAll(List<User> users);

void deleteAll(List<User> users);

}

**Скриншоты работы:**

