**Searching for a Specific User and Updating the User Information.**

**1. ) UsermanagerApplication.java**

package com.example.UserManager;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class UserManagerApplication {

public static void main(String[] args) {

SpringApplication.run(UserManagerApplication.class, args);

}

}

User.java

package com.example.UserManager.entities;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity // This tells Hibernate to make a table out of this class

@Table(name = "users")

public class User {

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

private Integer id;

private String name;

private String email;

private String password;

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

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 String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

@Override

public String toString() {

return (id.toString() + " " + name + " " + email + " " + password);

}

}

**UserNotfound.java**

package com.example.UserManager.exceptions;

public class UserNotFoundException extends RuntimeException {

private static final long serialVersionUID = 1L;

/\*

private final String userId;

public UserNotFoundException() {

super();

this.userId = id;

}

public String getUserId() {

return userId;

}

\*/

}

**UserRepository.java**

package com.example.UserManager.repositories;

import org.springframework.data.repository.CrudRepository;

import com.example.UserManager.entities.User;

public interface UserRepository extends CrudRepository<User, Integer> {

public User findByName(String name);

}

**UserService.java**

package com.example.UserManager.services;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.example.UserManager.entities.User;

import com.example.UserManager.exceptions.UserNotFoundException;

import com.example.UserManager.repositories.UserRepository;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public Iterable<User> GetAllUsers()

{

return userRepository.findAll();

}

public User GetUserByName(String name) {

User foundUser = userRepository.findByName(name);

return foundUser;

}

public User GetUserById(int id) {

Optional<User> foundUser = userRepository.findById(id);

//TODO: we need to decide how to handle a "Not Found" condition

if (!foundUser.isPresent()) {

System.out.println("user does not exist");

throw new UserNotFoundException();

}

return(foundUser.get());

}

public void UpdateUser(User usertoUpdate) {

userRepository.save(usertoUpdate);

}

}

Output Screenshot











