public class UserDto {

private int id;

@NotEmpty

@Size(min = 4, message = "Username must be min of 4 characters !!")

private String name;

@Email(message = "Email address is not valid !!")

@NotEmpty(message = "Email is required !!")

private String email;

@NotEmpty

@Size(min = 3, max = 10, message = "Password must be min of 3 chars and max of 10 chars !!")

private String password;

}

@RestControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(ResourceNotFoundException.class)

public ResponseEntity<ApiResponse> resourceNotFoundExceptionHandler(ResourceNotFoundException ex) {

String message = ex.getMessage();

ApiResponse apiResponse = new ApiResponse(message, false);

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

}

@ExceptionHandler(MethodArgumentNotValidException.class)

public ResponseEntity<Map<String, String>> handleMethodArgsNotValidException(MethodArgumentNotValidException ex) {

Map<String, String> resp = new HashMap<>();

ex.getBindingResult().getAllErrors().forEach((error) -> {

String fieldName = ((FieldError) error).getField();

String message = error.getDefaultMessage();

resp.put(fieldName, message);

});

return new ResponseEntity<Map<String, String>>(resp, HttpStatus.BAD\_REQUEST);

}

@ExceptionHandler(ApiException.class)

public ResponseEntity<ApiResponse> handleApiException(ApiException ex) {

String message = ex.getMessage();

ApiResponse apiResponse = new ApiResponse(message, true);

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

}

}

// POST-create user

@PostMapping("/save")

public ResponseEntity<UserDto> createUser(@Valid @RequestBody UserDto userDto) {

UserDto createUserDto = this.userService.createUser(userDto);

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

}

// PUT- update user

@PutMapping("/update/{userId}")

public ResponseEntity<UserDto> updateUser(@Valid @RequestBody UserDto userDto, @PathVariable("userId") Integer uid) {

UserDto updatedUser = this.userService.updateUser(userDto, uid);

return ResponseEntity.ok(updatedUser);

}

