|  |  |
| --- | --- |
|  | Министерство образования и науки Российской Федерации |
| Федеральное государственное автономное образовательное учреждение высшего образования |
| «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина» |
| Институт радиоэлектроники и информационных технологий – РтФ |
|  |
|  | |

**ОТЧЕТ**

О ВЫПОЛНЕНИИ

ЛАБОРАТОНОЙ РАБОТЫ № 3

Руководитель ст. пр. Н. А. Архипов

Студент гр. РИМ-230971 Е.В. Треглазов

Екатеринбург 2024

**Цель работы:** доработать простой REST сервис.

**Доработка простого REST сервиса**

Класс DateTimeUtil в пакете util

package ru.example.MySecondTestAppSpringBoot.util;  
  
import java.text.SimpleDateFormat;  
  
public class DateTimeUtil {  
 public static SimpleDateFormat getCustomFormat() {  
 return new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");  
 }  
}

Пакет service

ModifyResponseService:

package ru.example.MySecondTestAppSpringBoot.service;  
  
import org.springframework.stereotype.Service;  
import ru.example.MySecondTestAppSpringBoot.model.Response;  
  
@Service  
public interface ModifyResponseService {  
  
 Response modify(Response response);  
}

ModifyOperationUidResponseService:

package ru.example.MySecondTestAppSpringBoot.service;  
  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.stereotype.Service;  
import ru.don.MySecondTestAppSpringBoot.model.Response;  
  
import java.util.UUID;  
  
@Service  
@Qualifier("ModifyOperationUidResponseService")  
public class ModifyOperationUidResponseService implements ModifyResponseService {  
  
 @Override  
 public Response modify(Response response) {  
  
 UUID uuid = UUID.*randomUUID*();  
  
 response.setOperationUid(uuid.toString());  
  
 return response;  
  
 }  
}

ModifySystemTimeResponseService:

package ru.example.MySecondTestAppSpringBoot.service;  
  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.stereotype.Service;  
import ru.example.MySecondTestAppSpringBoot.model.Response;  
import ru.example.MySecondTestAppSpringBoot.util.DateTimeUtil;  
  
import java.util.Date;  
  
@Service  
@Qualifier("ModifySystemTimeResponseService")  
public class ModifySystemTimeResponseService implements ModifyResponseService {  
  
 @Override  
 public Response modify(Response response) {  
  
 response.setSystemTime(DateTimeUtil.*getCustomFormat*().format(new Date()));  
  
 return response;  
 }  
}

Перечисления Codes, ErrorMessages, ErrorCodes в пакете model

Codes:

package ru.example.MySecondTestAppSpringBoot.model;  
  
import com.fasterxml.jackson.annotation.JsonValue;  
  
public enum Codes {  
  
 *SUCCESS*("success"),  
 *FAILED*("failed");  
  
 private final String name;  
  
 Codes(String name) {  
 this.name = name;  
 }  
  
 @JsonValue  
 public String getName() {  
 return name;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
}

ErrorMessages:

package ru.example.MySecondTestAppSpringBoot.model;  
  
import com.fasterxml.jackson.annotation.JsonValue;  
  
public enum ErrorMessages {  
  
 *EMPTY*(""),  
 *VALIDATION*("Ошибка валидации"),  
 *UNSUPPORTED*("Произошла непредвиденная ошибка"),  
 *UNKNOWN*("Не поддерживаемая ошибка");  
  
 private final String description;  
  
 ErrorMessages(String description) {  
 this.description = description;  
 }  
  
 @JsonValue  
 public String getName() {  
 return description;  
 }  
}

ErrorCodes:

package ru.example.MySecondTestAppSpringBoot.model;  
  
import com.fasterxml.jackson.annotation.JsonValue;  
  
public enum ErrorCodes {  
  
 *EMPTY*(""),  
 *VALIDATION\_EXCEPTION*("ValidateionException"),  
 *UNKNOWN\_EXCEPTION*("UnknownException"),  
 *UNSUPPORTED\_EXCEPTION*("UnsupportedException");  
  
 private final String name;  
 ErrorCodes(String name) {  
 this.name = name;  
 }  
  
 @JsonValue  
 public String getName() {  
 return name;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
}

Модификация класса Response

package ru.example.MySecondTestAppSpringBoot.model;  
  
import lombok.Builder;  
import lombok.Data;  
  
@Data  
@Builder  
public class Response {  
 private String uid;  
 private String operationUid;  
 private String systemTime;  
 // private String code;  
 private Codes code;  
 // private String errorCode;  
 private ErrorCodes errorCode;  
 // private String errorMessage;  
 private ErrorMessages errorMessage;  
}

Доработка класса Request

package ru.example.MySecondTestAppSpringBoot.model;  
  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.validation.constraints.\*;  
@Data  
@Builder  
@NoArgsConstructor  
@AllArgsConstructor  
public class Request {  
 @NotBlank  
 @Size(max = 32)  
 private String uid;  
  
 @NotBlank  
 @Size(max = 32)  
 private String operationUid;  
  
 private Systems systemName;  
  
 @NotBlank  
 private String systemTime;  
  
 private String source;  
  
 @Min(1)  
 @Max(100000)  
 private int communicationId;  
  
 private int templateId;  
  
 private int productCode;  
  
 private int smsCode;  
  
 @Override  
 public String toString() {  
 return "{" +  
 "uid='" + uid + '\'' +  
 ", opertionUid='" + operationUid + '\'' +  
 ", systemName='" + systemName + '\'' +  
 ", systemTime='" + systemTime + '\'' +  
 ", source='" + source + '\'' +  
 ", communicationId=" + communicationId +  
 ", templateId=" + templateId +  
 ", productCode=" + productCode +  
 ", smsCode=" + smsCode +  
 '}';  
 }  
}

Добавление логирование

spring.application.name=MySecondTestAppSpringBoot  
server.port=8082  
logging.level.org.springframework=ERROR  
logging.level.ru.example.MySecondTestAppSpringBoot=DEBUG

Доработка класса MyController

package ru.example.MySecondTestAppSpringBoot.controller;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
import ru.example.MySecondTestAppSpringBoot.exception.\*;  
import ru.example.MySecondTestAppSpringBoot.model.\*;  
import ru.example.MySecondTestAppSpringBoot.service.ValidationService;  
import ru.example.MySecondTestAppSpringBoot.service.UnsupportedCodeValidation;  
import ru.example.MySecondTestAppSpringBoot.service.ModifyResponseService;  
import ru.example.MySecondTestAppSpringBoot.util.DateTimeUtil;  
import org.springframework.validation.BindingResult;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Qualifier;  
import javax.validation.Valid;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import lombok.extern.slf4j.Slf4j;  
  
@Slf4j  
@RestController  
public class MyController {  
 private final ValidationService validationService;  
 private final ModifyResponseService modifyResponseService;  
 private final UnsupportedCodeValidation unsupportedCodeValidation;  
  
 @Autowired  
 public MyController(ValidationService validationService, UnsupportedCodeValidation unsupportedCodeValidation,  
 @Qualifier("ModifySystemTimeResponseService") ModifyResponseService modifyResponseService) {  
 this.validationService = validationService;  
 this.unsupportedCodeValidation = unsupportedCodeValidation;  
 this.modifyResponseService = modifyResponseService;  
 }  
  
 @PostMapping(value = "/feedback")  
 public ResponseEntity<Response> feedback(@Valid @RequestBody Request request,  
 BindingResult bindingResult) {  
  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");  
  
 log.info("request: {}", request);  
  
 Response response = Response.builder()  
 .uid(request.getUid())  
 .operationUid(request.getOperationUid())  
 .systemTime(DateTimeUtil.*getCustomFormat*().format(new Date()))  
// .systemTime(simpleDateFormat.format(new Date()))  
 .code(Codes.*SUCCESS*)  
 .errorCode(ErrorCodes.*EMPTY*)  
 .errorMessage(ErrorMessages.*EMPTY*)  
 .build();  
  
 try {  
 validationService.isValid(bindingResult);  
 } catch (ValidationFailedException e) {  
 log.error("error: {}", e.getMessage());  
 response.setCode(Codes.FAILED);  
// response.setCode("failed");  
 response.setErrorCode(ErrorCodes.VALIDATION\_EXCEPTION);  
// response.setErrorCode("ValidationException");  
 response.setErrorMessage(ErrorMessages.VALIDATION);  
// response.setErrorMessage("Ошибка валидации");  
 return new ResponseEntity<>(response, HttpStatus.*BAD\_REQUEST*);  
 } catch (Exception e) {  
 log.error("error: {}", e.getMessage());  
 response.setCode(Codes.FAILED);  
// response.setCode("failed");  
 response.setErrorCode(ErrorCodes.UNKNOWN\_EXCEPTION);  
// response.setErrorCode("UnknownException");  
 response.setErrorMessage(ErrorMessages.UNKNOWN);  
// response.setErrorMessage("Произошла непредвиденная ошибка");  
 return new ResponseEntity<>(response, HttpStatus.*INTERNAL\_SERVER\_ERROR*);  
 }  
  
 try {  
 unsupportedCodeValidation.Valid(Integer.*parseInt*(request.getUid()));  
 } catch (UnsupportedCodeException e) {  
 log.error("error: {}", e.getMessage());  
 response.setCode(Codes.FAILED);  
// response.setCode("failed");  
 response.setErrorCode(ErrorCodes.UNSUPPORTED\_EXCEPTION);  
// response.setErrorCode("UnsupportedCodeException");  
 response.setErrorMessage(ErrorMessages.UNSUPPORTED);  
// response.setErrorMessage("uid = 123");  
 return new ResponseEntity<>(response, HttpStatus.*BAD\_REQUEST*);  
 }  
 modifyResponseService.modify(response);  
 log.info("response: {}", response);  
 return new ResponseEntity<>(response, HttpStatus.*OK*);  
 }  
}

**Реализация дополнительного функционала**

Добавьте вывод логов уровня INFO везде, где изменяется response и request таким образом чтобы по логам можно было отследить всю логику работы приложения.

@Slf4j  
@RestController  
public class MyController {  
 private final ValidationService validationService;  
 private final ModifyResponseService modifyResponseService;  
 private final UnsupportedCodeValidation unsupportedCodeValidation;  
  
 @Autowired  
 public MyController(ValidationService validationService, UnsupportedCodeValidation unsupportedCodeValidation,  
 @Qualifier("ModifySystemTimeResponseService") ModifyResponseService modifyResponseService) {  
 this.validationService = validationService;  
 this.unsupportedCodeValidation = unsupportedCodeValidation;  
 this.modifyResponseService = modifyResponseService;  
 }  
  
 @PostMapping(value = "/feedback")  
 public ResponseEntity<Response> feedback(@Valid @RequestBody Request request,  
 BindingResult bindingResult) {  
  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");  
  
 log.info("request: {}", request);  
  
 Response response = Response.builder()  
 .uid(request.getUid())  
 .operationUid(request.getOperationUid())  
 .systemTime(DateTimeUtil.*getCustomFormat*().format(new Date()))  
// .systemTime(simpleDateFormat.format(new Date()))  
 .code(Codes.*SUCCESS*)  
 .errorCode(ErrorCodes.*EMPTY*)  
 .errorMessage(ErrorMessages.*EMPTY*)  
 .build();  
  
 try {  
 validationService.isValid(bindingResult);  
 } catch (ValidationFailedException e) {  
 log.error("error: {}", e.getMessage());  
 response.setCode(Codes.*FAILED*);  
// response.setCode("failed");  
 response.setErrorCode(ErrorCodes.*VALIDATION\_EXCEPTION*);  
// response.setErrorCode("ValidationException");  
 response.setErrorMessage(ErrorMessages.*VALIDATION*);  
// response.setErrorMessage("Ошибка валидации");  
 return new ResponseEntity<>(response, HttpStatus.*BAD\_REQUEST*);  
 } catch (Exception e) {  
 log.error("error: {}", e.getMessage());  
 response.setCode(Codes.*FAILED*);  
// response.setCode("failed");  
 response.setErrorCode(ErrorCodes.*UNKNOWN\_EXCEPTION*);  
// response.setErrorCode("UnknownException");  
 response.setErrorMessage(ErrorMessages.*UNKNOWN*);  
// response.setErrorMessage("Произошла непредвиденная ошибка");  
 return new ResponseEntity<>(response, HttpStatus.*INTERNAL\_SERVER\_ERROR*);  
 }  
  
 try {  
 unsupportedCodeValidation.Valid(Integer.*parseInt*(request.getUid()));  
 } catch (UnsupportedCodeException e) {  
 log.error("error: {}", e.getMessage());  
 response.setCode(Codes.*FAILED*);  
// response.setCode("failed");  
 response.setErrorCode(ErrorCodes.*UNSUPPORTED\_EXCEPTION*);  
// response.setErrorCode("UnsupportedCodeException");  
 response.setErrorMessage(ErrorMessages.*UNSUPPORTED*);  
// response.setErrorMessage("uid = 123");  
 return new ResponseEntity<>(response, HttpStatus.*BAD\_REQUEST*);  
 }  
 modifyResponseService.modify(response);  
 log.info("response: {}", response);  
 return new ResponseEntity<>(response, HttpStatus.*OK*);  
 }  
}

Добавьте вывод логов уровня ERROR в те места программы, где выбрасываются исключения, а также добавить сообщения, которые возникают при ошибке в bindingResult.

@Slf4j  
@RestController  
public class MyController {  
 private final ValidationService validationService;  
 private final ModifyResponseService modifyResponseService;  
 private final UnsupportedCodeValidation unsupportedCodeValidation;  
  
 @Autowired  
 public MyController(ValidationService validationService, UnsupportedCodeValidation unsupportedCodeValidation,  
 @Qualifier("ModifySystemTimeResponseService") ModifyResponseService modifyResponseService) {  
 this.validationService = validationService;  
 this.unsupportedCodeValidation = unsupportedCodeValidation;  
 this.modifyResponseService = modifyResponseService;  
 }  
  
 @PostMapping(value = "/feedback")  
 public ResponseEntity<Response> feedback(@Valid @RequestBody Request request,  
 BindingResult bindingResult) {  
  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");  
  
 log.info("request: {}", request);  
  
 Response response = Response.builder()  
 .uid(request.getUid())  
 .operationUid(request.getOperationUid())  
 .systemTime(DateTimeUtil.*getCustomFormat*().format(new Date()))  
// .systemTime(simpleDateFormat.format(new Date()))  
 .code(Codes.*SUCCESS*)  
 .errorCode(ErrorCodes.*EMPTY*)  
 .errorMessage(ErrorMessages.*EMPTY*)  
 .build();  
  
 try {  
 validationService.isValid(bindingResult);  
 } catch (ValidationFailedException e) {  
 log.error("error: {}", e.getMessage());  
 response.setCode(Codes.*FAILED*);  
// response.setCode("failed");  
 response.setErrorCode(ErrorCodes.*VALIDATION\_EXCEPTION*);  
// response.setErrorCode("ValidationException");  
 response.setErrorMessage(ErrorMessages.*VALIDATION*);  
// response.setErrorMessage("Ошибка валидации");  
 return new ResponseEntity<>(response, HttpStatus.*BAD\_REQUEST*);  
 } catch (Exception e) {  
 log.error("error: {}", e.getMessage());  
 response.setCode(Codes.*FAILED*);  
// response.setCode("failed");  
 response.setErrorCode(ErrorCodes.*UNKNOWN\_EXCEPTION*);  
// response.setErrorCode("UnknownException");  
 response.setErrorMessage(ErrorMessages.*UNKNOWN*);  
// response.setErrorMessage("Произошла непредвиденная ошибка");  
 return new ResponseEntity<>(response, HttpStatus.*INTERNAL\_SERVER\_ERROR*);  
 }  
  
 try {  
 unsupportedCodeValidation.Valid(Integer.*parseInt*(request.getUid()));  
 } catch (UnsupportedCodeException e) {  
 log.error("error: {}", e.getMessage());  
 response.setCode(Codes.*FAILED*);  
// response.setCode("failed");  
 response.setErrorCode(ErrorCodes.*UNSUPPORTED\_EXCEPTION*);  
// response.setErrorCode("UnsupportedCodeException");  
 response.setErrorMessage(ErrorMessages.*UNSUPPORTED*);  
// response.setErrorMessage("uid = 123");  
 return new ResponseEntity<>(response, HttpStatus.*BAD\_REQUEST*);  
 }  
 modifyResponseService.modify(response);  
 log.info("response: {}", response);  
 return new ResponseEntity<>(response, HttpStatus.*OK*);  
 }

В классе Request измените тип поля systemName с типа String на тип созданного самостоятельно перечисления Systems. Элементами перечисления сделайте следующие значения ERP («Enterprise Resource Planning»), CRM («Customer Relationship Management»), WMS (Warehouse Management System).

package ru.example.MySecondTestAppSpringBoot.model;  
  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.validation.constraints.\*;  
@Data  
@Builder  
@NoArgsConstructor  
@AllArgsConstructor  
public class Request {  
 @NotBlank  
 @Size(max = 32)  
 private String uid;  
  
 @NotBlank  
 @Size(max = 32)  
 private String operationUid;  
  
 private Systems systemName;  
  
 @NotBlank  
 private String systemTime;  
  
 private String source;  
  
 @Min(1)  
 @Max(100000)  
 private int communicationId;  
  
 private int templateId;  
  
 private int productCode;  
  
 private int smsCode;  
  
 @Override  
 public String toString() {  
 return "{" +  
 "uid='" + uid + '\'' +  
 ", opertionUid='" + operationUid + '\'' +  
 ", systemName='" + systemName + '\'' +  
 ", systemTime='" + systemTime + '\'' +  
 ", source='" + source + '\'' +  
 ", communicationId=" + communicationId +  
 ", templateId=" + templateId +  
 ", productCode=" + productCode +  
 ", smsCode=" + smsCode +  
 '}';  
 }  
}

**Ответы на вопросы**

4 вариант

2. Что такое CRM система? Приведите примеры.

CRM (Customer Relationship Management) система – это программное обеспечение, которое помогает компаниям управлять отношениями с клиентами, автоматизировать процессы продаж и маркетинга, а также улучшать обслуживание клиентов.

Примеры CRM систем: Salesforce, HubSpot, Zoho CRM, Bitrix24.

4. Какие библиотеки логгирования бывают в Spring?

Logback – это библиотека логгирования, которая является преемником Log4j и предоставляет более высокую производительность и гибкость.

Log4j – это одна из самых распространенных библиотек логгирования в Java, которая позволяет настраивать уровни логгирования для различных компонентов приложения.

JUL (Java Util Logging) – это стандартная библиотека логгирования в Java, которая включена в JDK.

6. Что означает аннотация @Qualifier?

Аннотация @Qualifier используется в Spring для разрешения конфликтов при инъекции зависимостей, когда есть несколько бинов с одинаковым типом и нужно указать, какой именно бин должен быть использован. Аннотация @Qualifier применяется вместе с именем бина или другими значениями, которые помогают идентифицировать нужный бин.

**Ссылка на репозиторий git**:

https://github.com/kuk86kuk/java\_ufru\_2024