INT204 OASIP-BackEnd SSA-5

สมาชิกกลุ่ม

นาย ทรงสิจ สินนุรักษ์ 63130500043

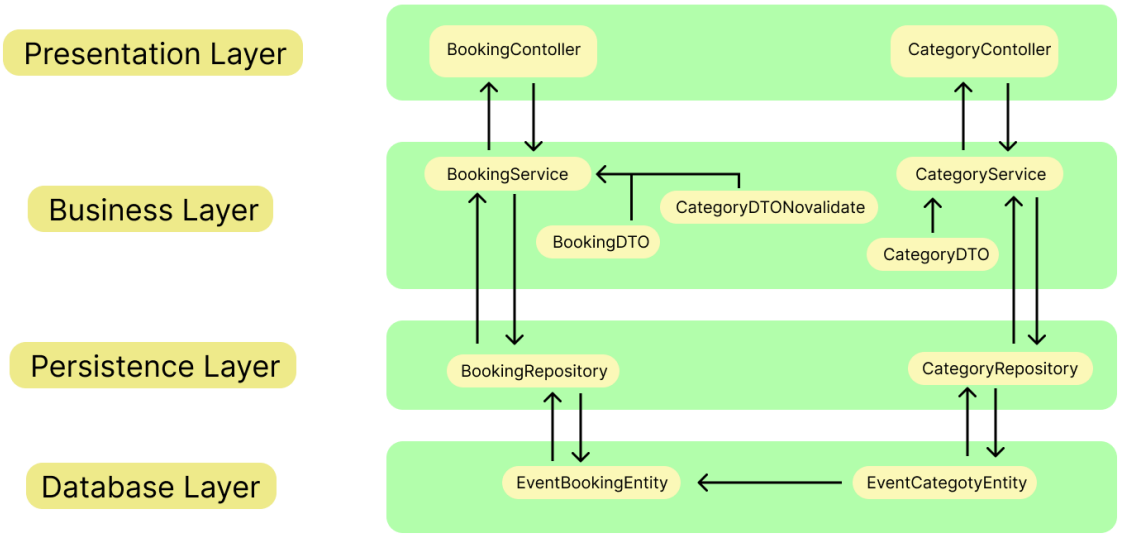
นาย ชนาธิป เอมเปีย 63130500015

นาย ปรมัตถ์ เพ็ชรอินทร์ 63130500078

End-point

Url	HTTP verb	Description
/api/bookings	GET	Get all EventBooking and filter by “startTime”
/api/bookings/{BookingId}	GET	Get EventBooking with BookingId
/api/bookings/sortByCategory	GET	Get EventBooking filter by “category”
/api/bookings/sortByPast	GET	Get EventBooking filter by past of date
/api/bookings/sortBySpecify	GET	Get EventBooking filter by current date
/api/bookings	POST	Add a new EventBooking
/api/bookings/{BookingId}	PUT	Update an EventBooking with BookingId
/api/bookings/{bookingId}	DELETE	Delete an EventBooking with BookingId
/api/categories	GET	Get all EventCategory
/api/categories/{CategoryId}	GET	Get EventCategory with CategoryId
/api/categories/{CategoryId}	PUT	Update an EventCategory with CategoryId

Layer Architecture



DTO Class

BookingDTO

```java
package sit.integrate.oasip.DTO;

import lombok.*;

import javax.validation.Valid;
import javax.validation.constraints.*;
import java.time.LocalDateTime;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@ToString
public class BookingDTO {
    private Integer id;
    @NotNull(message = "Name shouldn't be null")
    @NotEmpty(message = "Name shouldn't be empty")
    @Size(max = 100,message = "Your Name have length more than 100 character")
    @NotBlank(message = "Name shouldn't be null/empty")
    private String bookingName;

    @Email(message = "Email doesn't follow format")
    @NotBlank(message = "Email shouldn't be null/empty")
    @NotNull(message = "Email shouldn't be null")
    @NotEmpty(message = "Email shouldn't be empty")
    @Size(max = 100,message = "Your Email have length more than 100 character")
    private String bookingEmail;

    @NotNull(message = "Category is null")
    private CategoryDTONoValidate category;

    @Future(message = "Can't Change to Date and Time in Past")
    @NotNull(message = "Date and Time shouldn't be null")
    private LocalDateTime startTime;

    @Max(value = 480,message = "Duration have more than 480 minute")
    @Min(value = 1,message = "Duration have less 1 minute")
    @NotNull(message = "Duration shouldn't be null")
    private Integer bookingDuration;

    @Size(max = 500,message = "Your Note have length more than 500 character")
    private String eventNote;

    public String getStartTime(){
        return startTime.toString();
    }

    public void setStartTime(String startTime) {
        this.startTime = LocalDateTime.parse(startTime);
    }
}
```

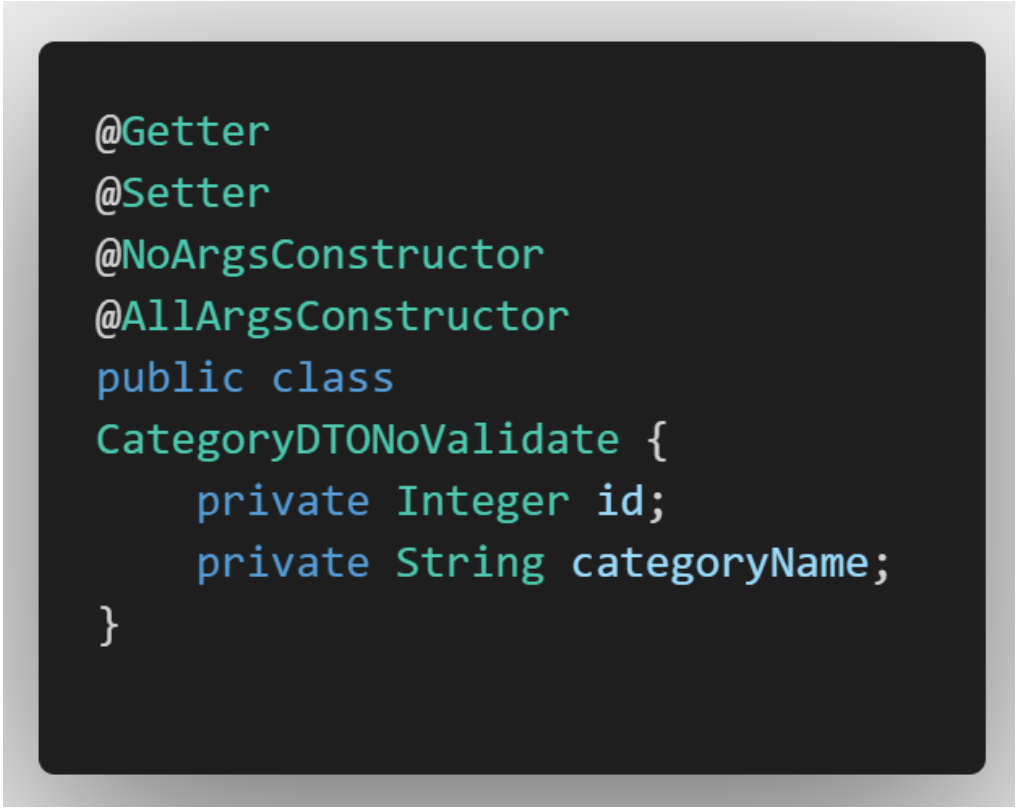
ตัว BookingDTO มีการใส่ validation ต่าง ๆ จาก Dependency Spring-boot เช่น การใส่ Size เพื่อกำหนดจำนวนคำของค่า String ใส่ NotNull เพื่อไม่ให้ค่านั้นเก็บค่าว่างปล่าว เป็นต้น

CategoryDTO

```java
package sit.integrate.oasip.DTO;

import lombok.*;
import org.hibernate.validator.constraints.UniqueElements;
import javax.validation.constraints.*;


@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class CategoryDTO {
    private Integer id;
    @NotBlank(message = "Category name shouldn't null/empty")
    @NotEmpty(message = "Category name shouldn't empty")
    @NotNull(message = "Category name shouldn't null")
    @Size(max = 100,message = "Category Name have length more than 100 character")
    private String categoryName;

    @Size(max = 500,message = "Category description have length more than 500 character")
    private String description;

    @Max(value = 480,message = "Duration have more than 480 minute")
    @Min(value = 1,message = "Duration have less 1 minute")
    @NotNull(message = "Duration shouldn't be null")
    private Integer duration;

}
```

ตัว CategoryDTO มีการใส่ validation ต่าง ๆ จาก Dependency Spring-boot เช่น การใส่ Size เพื่อกำหนดจำนวนคำของค่า String ใส่ NotNull เพื่อไม่ให้ค่านั้นเก็บค่าว่างปล่าว เป็นต้น

CategoryDTONoValidate



```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class
CategoryDTONoValidate {
    private Integer id;
    private String categoryName;
}
```

มีไว้เพื่อให้ BookingDTO ที่ใช้ในรับหรือส่งข้อมูลในรูปแบบ Json ในตัวของ category ไม่จำเป็นต้องส่งตัวของ Duration และ Description มาด้วย และเลี่ยงการ validate จากการส่งข้อมูลมา

BookingController , BookingService and BookingRepository Class

(BookingService)

```
public List<BookingDTO> getBookings(int page, int pageSize, String sort){
    List<EventBooking> bookingList = repository.findAll(PageRequest.of(page,pageSize,Sort.by(Sort.
Direction.DESC,sort))).getContent();
    return listMapper.mapList(bookingList, BookingDTO.class, modelMapper);
}
```

เป็น method ที่หาค่าทั้งหมดของ Entity EventBooking

(BookingController)

```
@GetMapping("")
    public ResponseEntity<List<BookingDTO>> getAllBooking(
            @RequestParam(defaultValue = "0") int page,
            @RequestParam(defaultValue = "5") int pageSize,
            @RequestParam(defaultValue = "startTime") String sort){
        return ResponseEntity.ok(service.getBookings(page,pageSize,sort));
    }
```

เป็น method ที่ return ค่าจาก method getAllBooking ของ BookingService และแสดงค่าออกมาตาม endpoint ที่กำหนดไว้

(BookingService)

```
public BookingDTO getBookingId(Integer bookingId){
    EventBooking booking = repository.findById(bookingId)
            .orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT_FOUND,"Booking id "
+bookingId+" does not exits."));
    return modelMapper.map(booking, BookingDTO.class);
}
```

เป็น method ที่รับค่า Integer bookingId เพื่อมห้ใช้คำสั่ง findById ในการหา EventBooking ด้วยการใช้ค่า booingId ที่รับมา

(BookingController)

```
@GetMapping("/{BookingId}")
    public ResponseEntity<BookingDTO> getBooking(@PathVariable Integer BookingId){
        return ResponseEntity.ok(service.getBookingId(BookingId));
    }
```

เป็น method ที่ return ค่าจาก method getBookingId ของ BookingService และแสดงค่าออกมาตาม endpoint ที่กำหนดไว้

(BookingRepository)

```
List<EventBooking> findAllByCategoryOrderByStartTimeDesc(Pageable page, EventCategory category);
```

เป็น Query SQL ในรูปแบบของ Spring Boot ที่หาค่าทุกตัวใน EventBooking โดยใช้ค่า Category โดยรับและใช้แค่ categoryId ในการ Run Query และเรียงลำดับจากค่า StartTime ในอนาคต ไปยังอดีต

(BookingService)

```
public List<BookingDTO> getBookingCategory(
            int page,
            int pageSize,
            EventCategory category){
        List<EventBooking> bookingList = repository.findAllByCategoryOrderByStartTimeDesc(PageRequest
.of(page,pageSize),category);
        return listMapper.mapList(bookingList, BookingDTO.class,modelMapper);
    }
```

เป็น method ที่เรียกใช้คำสั่งจากตัว repository ที่ใช้หาตัว EventBooking ที่มีการ filter โดยใช้ค่า category ในการหา

(BookingController)

```
@GetMapping("/sortByCategory")
    public ResponseEntity<List<BookingDTO>> getBookingByCategory(
            @RequestParam(defaultValue = "0") int page,
            @RequestParam(defaultValue = "5") int pageSize,
            @RequestParam EventCategory category){
        return ResponseEntity.ok(service.getBookingCategory(page,pageSize,category));
    }
```

เป็น method ที่ return ค่าจาก method getBookingCategory ของ BookingService และแสดงค่าออกมาตาม endpoint ที่กำหนดไว้

(BookingRepository)

```
List<EventBooking> findAllByStartTimeLessThanOrderByStartTimeDesc(Pageable page,LocalDateTime
localDateTime);
```

เป็น Query SQL ในรูปแบบของ Spring Boot ที่หาค่าทุกตัวใน EventBooking โดยที่ต้องมีค่า StartTime น้อยกว่าเวลาในปัจจุบันโดยเรียงลำดับจาก StartTime ที่ใกล้ปัจจุบันที่สุดไปยังอดีต

(BookingService)

```
public List<BookingDTO> getBookingSortPast(int page, int pageSize,LocalDateTime localDateTime){
        List<EventBooking> bookingList = repository.findAllByStartTimeLessThanOrderByStartTimeDesc(
PageRequest.of(page,pageSize),localDateTime);
        return listMapper.mapList(bookingList, BookingDTO.class, modelMapper);
    }
```

เป็น method ที่เรียกใช้คำสั่งจากตัว repository ที่ใช้หาตัว EventBooking โดยใช้ค่า LocalDateTime ในการหา

(BookingController)

```
@GetMapping("/sortByPast")
    public ResponseEntity<List<BookingDTO>> getAllBookingByPast(
            @RequestParam(defaultValue = "0") int page,
            @RequestParam(defaultValue = "5") int pageSize
    ){
        return ResponseEntity.ok(service.getBookingSortPast(page,pageSize,LocalDateTime.now()));
    }
```

เป็น method ที่ return ค่าจาก method ของ BookingService และแสดงค่าออกมาใน endpoint ที่กำหนดไว้

(BookingRepository)

```
List<EventBooking> findAllByStartTimeBetweenOrderByStartTimeAsc(Pageable page,LocalDateTime startdate
,LocalDateTime enddate);
```

เป็น Query SQL ในรูปแบบของ Spring Boot ที่หาค่าทุกตัวใน EventBooking โดยต้องมีค่าระหว่าง startDate (เริ่มต้นวัน) และ endDate (จบวัน) เรียงลำดับจากค่า StartTime น้อยไปมาก

(BookingService)

```
public List<BookingDTO> getBookingWithSpecify(
            int page,
            int pageSize,
            String startdate,
            String enddate){
        List<EventBooking> bookingList = repository.findAllByStartTimeBetweenOrderByStartTimeAsc(
                PageRequest.of(page,pageSize),
                LocalDateTime.parse(startdate),
                LocalDateTime.parse(enddate));
        return listMapper.mapList(bookingList, BookingDTO.class,modelMapper);
    }
```

เป็น method ที่เรียกใช้คำสั่งจากตัว repository ที่ใช้หาตัว EventBooking โดยใช้ค่า LocalDateTime 2 ตัวในการหา

(BookingController)

```
@GetMapping("/sortByDay")
    public ResponseEntity<List<BookingDTO>> getBookingBySpecify(
            @RequestParam(defaultValue = "0") int page,
            @RequestParam(defaultValue = "5") int pageSize,
            @RequestParam String date){
        return ResponseEntity.ok(service.getBookingWithSpecify(
                page,
                pageSize,
                date+"T00:00",
                date+"T23:59"
        ));
    }
```

เป็น method ที่ return ค่าจาก method getBookingWithSpecify ของ BookingService และแสดงค่าออกมาตาม endpoint ที่กำหนดไว้

(BookingService)

```
public EventBooking CreateBooking(BookingDTO newBooking){
        newBooking.setBookingName(newBooking.getBookingName().trim());
        newBooking.setBookingEmail(newBooking.getBookingEmail().trim());
        newBooking.setEventNote(newBooking.getEventNote().trim());
        EventBooking booking = modelMapper.map(newBooking,EventBooking.class);
        return repository.saveAndFlush(booking);
    }
```

เป็น method ที่ทำการสร้างตัว EventBooking ขึ้นมาใหม่โดยมีการ set BookingName, Bookingemail, EventNote ไม่ให้มีช่องว่างหน้าและหลัง

(BookingController)

```
@PostMapping("")
    @ResponseStatus(HttpStatus.CREATED)
    public ResponseEntity<BookingDTO> AddBooking(@Valid @RequestBody BookingDTO newBooking){
        EventBooking eventBooking=service.CreateBooking(newBooking);
        return new ResponseEntity<>(modelMapper.map(eventBooking,BookingDTO.class),HttpStatus.CREATED
);
    }
```

เป็น method ที่ return ค่าจาก method CreateBooking ของ BookingService และแสดงค่าออกมาตาม endpoint ที่กำหนดไว้

(BookingService)

```
public EventBooking UpdateBooking(Integer bookingId,BookingDTO updateBooking) throws
BookingException {
        BookingDTO oldBooking = getBookingId(bookingId);
        List<String> errors=new ArrayList<>();
        if(!oldBooking.getBookingName().equals(updateBooking.getBookingName())){
            errors.add("The bookingName can't change");
            if(!oldBooking.getBookingEmail().equals(updateBooking.getBookingEmail())){
                errors.add("The bookingEmail can't change");
            }
            throw new BookingException(errors.toString());
        }
        updateBooking.setBookingName(updateBooking.getBookingName().trim());
        updateBooking.setBookingEmail(updateBooking.getBookingEmail().trim());
        updateBooking.setEventNote(updateBooking.getEventNote().trim());
        EventBooking booking = repository.findById(bookingId).map(b->mapBooking(modelMapper.map(b,
BookingDTO.class),updateBooking))
                .orElseGet(()->{
                    updateBooking.setId(bookingId);
                    return modelMapper.map(updateBooking,EventBooking.class);
                });
        return repository.saveAndFlush(booking);
    }
```

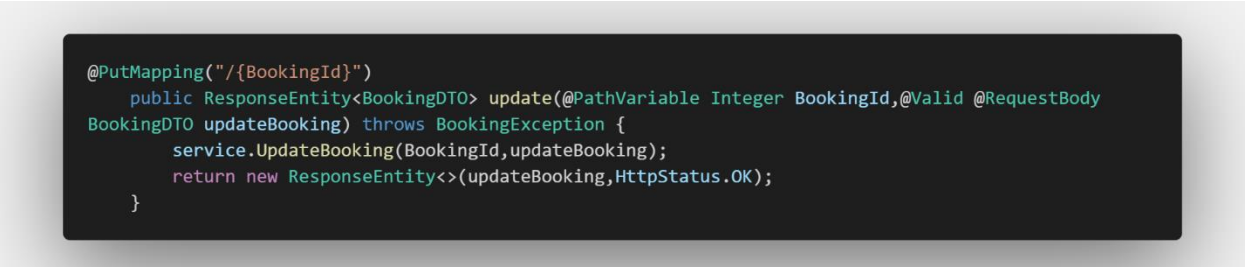
เป็น method ที่ทำการเข้าไปแก้ไขค่าของตัว EventBooking โดยการรับค่า bookingId เพื่อเข้าไปแก้ไขตัว EventBooking ตัวนั้น ๆ โดยได้มีการ check error ไว้ว่าถ้าไปแก้ไขตัว BookigName และ BookingEmail จะไม่สามารถแก้ไขได้และขึ้น error Exception หามีการแก้ไขโดยตรง และอัปเดตตัวที่แก้ไขเข้าไปแทนที่ตัวเก่าโดยมีการ set BookingName, Bookingemail, EventNote ไม่ให้มีช่องว่างหน้าและหลัง

(mapBooking)

```
private EventBooking mapBooking(BookingDTO oldBooking,BookingDTO newBooking){
        oldBooking=newBooking;
        return modelMapper.map(oldBooking,EventBooking.class);
    }
```

เป็น method ที่รับ Booking 2 ตัว ที่เป็นข้อมูลเก่าและที่เป็นข้อมูลใหม่ และกำหนดให้ตัวเก่ามีค่าเท่าตัวใหม่ ใช้ใน method UpdateBooking โดยเฉพาะ

(BookingController)



```
@PutMapping("/{BookingId}")
    public ResponseEntity<BookingDTO> update(@PathVariable Integer BookingId,@Valid @RequestBody
BookingDTO updateBooking) throws BookingException {
        service.UpdateBooking(BookingId,updateBooking);
        return new ResponseEntity<>(updateBooking,HttpStatus.OK);
    }
```

เป็น method ที่ return ค่าจาก method UpdateBooking ของ BookingService และแสดงค่าออกมาตาม endpoint ที่กำหนดไว้

(BookingService)

```
public void DeleteBooking(Integer BookingId){
        repository.deleteById(BookingId);
    }
```

เป็น method ที่ทำการลบ EventBooking โดยรับค่า BookingId เพื่อลบ EventBooking ตัวนั้น

(BookingController)

```
@DeleteMapping("/{bookingId}")
public void deleteBooking(@PathVariable Integer bookingId){
    service.DeleteBooking(bookingId);
}
```

เป็น method ที่ return ค่าจาก method DeleteBooking ของ BookingService และแสดงค่าออกมาใน endpoint ที่กำหนดไว้

CategoryController and CategoryService class

(CategoryService)

```
public List<CategoryDTO> getCategories(){
        List<EventCategory> categoryList = repository.findAll();
        return listMapper.mapList(categoryList, CategoryDTO.class, modelMapper);
    }
```

เป็น method ที่หาค่าทั้งหมดของ EventCategory

(CategoryController)

```
@GetMapping("")
    public ResponseEntity<List<CategoryDTO>> getAllCategory(){
        return ResponseEntity.ok(service.getCategories());
    }
```

เป็น method ที่ return ค่าจาก method ของ CategoryService และแสดงค่าออกมาใน endpoint ที่กำหนดไว้

(CategoryService)

```
public CategoryDTO getCategoryById(Integer categoryId){
        EventCategory category =repository.findById(categoryId)
                .orElseThrow(()->new ResponseStatusException(
                        HttpStatus.NOT_FOUND,"Category id"+ categoryId +" Does Not Exist !!!"
                ));
        return modelMapper.map(category,CategoryDTO.class);
    }
```

เป็น method ที่หาค่า EventCategory โดยใช้ค่า categoryId เพื่อหา EventCategory ตัวนั้น

(CategoryController)

```
@GetMapping("/{CategoryId}")
    public ResponseEntity<CategoryDTO> getCategoryById(@PathVariable Integer CategoryId){
        return ResponseEntity.ok(service.getCategoryById(CategoryId));
    }
```

เป็น method ที่ return ค่าจาก method ของ CategoryService และแสดงค่าออกมาใน endpoint ที่กำหนดไว้

(CategoryService)

```
public EventCategory UpdateCategory(Integer CategoryId, CategoryDTO updateCategory){
        EventCategory category = repository.findById(CategoryId).map(c->mapCategory(modelMapper.map
(c,CategoryDTO.class),updateCategory))
                .orElseGet(()->{
                    updateCategory.setId(CategoryId);
                    return modelMapper.map(updateCategory,EventCategory.class);
                });
        updateCategory.setCategoryName(updateCategory.getCategoryName().trim());
        updateCategory.setDescription(updateCategory.getDescription().trim());
        return repository.saveAndFlush(category);
    }
```

เป็น method ที่เข้าไปแก้ไขตัว EventCategory โดยใช้ค่า CategoryId เพื่อเข้าไปแก้ไข EventCategory ตัวนั้น และอัปเดตค่าเข้าไปแทน EventCategory ตัวเก่า และมีการ set CategoryName และ Description ไม่ให้มีช่องว่างตัวข้างหน้าและข้างหลัง

(mapCategory)

```
private EventCategory mapCategory(CategoryDTO oldCategory,CategoryDTO newCategory){
        oldCategory=newCategory;
        return modelMapper.map(oldCategory,EventCategory.class);
    }
```

เป็น method ที่รับ Category 2 ตัว ที่เป็นข้อมูลเก่าและที่เป็นข้อมูลใหม่ และกำหนดให้ตัวเก่ามีค่าเท่าตัวใหม่ ใช้ใน method UpdateCategory โดยเฉพาะ

(CategoryController)

```
@PutMapping("/{CategoryId}")
    public ResponseEntity<CategoryDTO> Update(@PathVariable Integer CategoryId, @Valid @RequestBody
CategoryDTO updateCategory){
        service.UpdateCategory(CategoryId,updateCategory);
        return new ResponseEntity<>(updateCategory,HttpStatus.OK);
    }
```

เป็น method ที่ return ค่าจาก method ของ CategoryService และแสดงค่าออกมาใน endpoint ที่กำหนดไว้

Exeption Class

(BookingException)

```java
public class BookingException extends Exception {
    public BookingException(String message) {
        super(message);
    }
}
```

เป็น Class Exception Handle

(ShowExeption)

```java
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class ShowException{
    private Integer statusCode;
    private String error;
    private Map<String,String> errorMessage;
}
```

เป็น Class ที่เอาไว้ใช้เพื่อให้แสดงข้อมูลของ Exception ตามเราต้องการ

(ApplicationExceptionHandler)

```java
@ResponseStatus(HttpStatus.BAD_REQUEST)
    @ExceptionHandler(value = MethodArgumentNotValidException.class)
    public ShowException handleValidationException(MethodArgumentNotValidException ex) {
        ShowException errors=new ShowException();
        Map<String, String> errorsMap = new HashMap<>();
        errors.setStatusCode(400);
        errors.setError("BAD REQUEST");
        ex.getBindingResult().getAllErrors().forEach((error) -> {
            String fieldName = ((FieldError) error).getField();
            String errorMessage = error.getDefaultMessage();
            errorsMap.put(fieldName,errorMessage);
        });
        errors.setErrorMessage(errorsMap);
        return errors;
    }
```

เป็น Exception method ที่แสดงเมื่อมีการ validate ไม่ผ่านตามเงื่อนไขใน DTO Class

```java
@ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
@ExceptionHandler(BookingException.class)
public ShowException handleBusinessException(BookingException ex) {
    ShowException errors=new ShowException();
    errors.setStatusCode(500);
    errors.setError("INTERNAL SERVER ERROR");
    Map<String, String> errorMap = new HashMap<>();
    errorMap.put("Message", ex.getMessage());
    errors.setErrorMessage(errorMap);
    return errors;
}
```

เป็น Exeption metod ที่แสดงเมื่อแก้ไขข้อมูลที่ไม่อนุญาตให้แก้ไข หรือมีการแก้ไขข้อมูลโดยตรง เช่น BookingName BookingEmail