

MovieBooking.java MovieBookingServiceImpl.java

```
1 package com.infy.moviebooking.entity;
2
3 import java.time.LocalDate;
4
5 @Entity
6 @Table(name="movie_booking")
7 public class MovieBooking {
8
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    private Integer bookingId;
12    private String movieName;
13    private String screenName;
14    private LocalDate showDate;
15    private Integer noOfSeats;
16    private Double bookingAmount;
17    private String paymentType;
18    private Long customerPhoneNo;
19
20    public Integer getBookingId() {
21        return bookingId;
22    }
23
24    public String getMovieName() {
25        return movieName;
26    }
27
28    public String getScreenName() {
29        return screenName;
30    }
31
32    public LocalDate getShowDate() {
33        return showDate;
34    }
35
36    public Integer getNoOfSeats() {
37        return noOfSeats;
38    }
39}
```

```
1 package com.infy.moviebooking.dto;
2 import java.time.LocalDate;
11 public class MovieBookingDTO {
12     private Integer bookingid;
13
14     @NotNull(message="{movie.name.absent}")
15     private String movieName;
16
17     @NotNull(message="{movie.screen.absent}")
18     @Pattern(regexp = "(Sapphire|Turquoise|Rhombus)", message = "{movie.screen.invalid}")
19     private String screenName;
20
21     @NotNull(message="{movie.show.absent}")
22     @FutureOrPresent(message = "{movie.show.invalid}")
23     private LocalDate showDate;
24
25
26     @NotNull(message = "{movie.noOfSeats.absent}")
27     @Min(value = 1,message = "{movie.noOfSeats.invalid}")
28     @Max(value = 5,message = "{movie.noOfSeats.invalid}")
29     private Integer noOfSeats;
30
31
32     private Double bookingAmount;
33     private String paymentType;
34     private Long customerPhoneNo;
35     public Integer getBookingid() {
36         return bookingid;
37     }
38     public void setBookingId(Integer bookingid) {
39         this.bookingid = bookingid;
40     }
41     public String getMovieName() {
42         return movieName;
43     }
44     public void setMovieName(String movieName) {
```



```
80 public void setCustomerPhoneNo(Long customerPhoneNo) {
81     this.customerPhoneNo = customerPhoneNo;
82 }
83
84 public static MovieBooking prepareEntity(MovieBookingDTO movieBookingDTO) {
85     MovieBooking movieBooking = new MovieBooking();
86     movieBooking.setMovieName(movieBookingDTO.getMovieName());
87     movieBooking.setScreenName(movieBookingDTO.getScreenName());
88     movieBooking.setShowDate(movieBookingDTO.getShowDate());
89     movieBooking.setNoOfSeats(movieBookingDTO.getNoOfSeats());
90     movieBooking.setBookingAmount(movieBookingDTO.getBookingAmount());
91     movieBooking.setPaymentType(movieBookingDTO.getPaymentType());
92     movieBooking.setCustomerPhoneNo(movieBookingDTO.getCustomerPhoneNo());
93     return movieBooking;
94 }
95
96
97
98 public static MovieBookingDTO prepareDTO( MovieBooking movieBooking) {
99     MovieBookingDTO movieBookingDTO = new MovieBookingDTO();
100     movieBookingDTO.setBookingId(movieBooking.getBookingId());
101     movieBookingDTO.setMovieName(movieBooking.getMovieName());
102     movieBookingDTO.setScreenName(movieBooking.getScreenName());
103     movieBookingDTO.setShowDate(movieBooking.getShowDate());
104     movieBookingDTO.setNoOfSeats(movieBooking.getNoOfSeats());
105     movieBookingDTO.setBookingAmount(movieBooking.getBookingAmount());
106     movieBookingDTO.setPaymentType(movieBooking.getPaymentType());
107     movieBookingDTO.setCustomerPhoneNo(movieBooking.getCustomerPhoneNo());
108     return movieBookingDTO;
109 }
110
111
112
113 }
114
```

```

1 package com.infy.moviebooking.repository;
2
3 import java.time.LocalDate;
4
5
6
7
8
9
10
11
12 public interface MovieBookingRepository extends CrudRepository<MovieBooking, Integer> {
13     @Query("select m from MovieBooking m where m.customerPhoneNo=?1 and m.showDate=?2")
14     List<MovieBooking> getBookingDetails(Long custmoerPhoneNo, LocalDate showDate );
15     List<MovieBooking> findByScreenName(String screenName);
16 }
17
    
```


MovieBooking.java MovieBookingServiceImpl.java MovieBookingDTO.java MovieBookingRepository.java

```
1 package com.infy.moviebooking.service;
2
3 import java.util.ArrayList;
4
5
6
7
8
9
10
11
12
13
14
15
16
17 @Service(value="movieBookingService")
18 @Transactional
19 public class MovieBookingServiceImpl implements MovieBookingService {
20
21     @Autowired
22     private MovieBookingRepository movieBookingRepository;
23
24     @Override
25     public MovieBookingDTO bookMovie(MovieBookingDTO movieBookingDTO) throws MovieBookingException {
26
27         MovieBookingValidator.validate(movieBookingDTO);
28         List<MovieBooking> movieBookingList = movieBookingRepository.getBookingDetails(movieBookingDTO.getCustomerPhoneNo(),
29             movieBookingDTO.getShowDate());
30
31         if(Boolean.FALSE.equals(movieBookingList.isEmpty())){
32
33             throw new MovieBookingException("MovieBookingService.BOOKING_EXISTS");
34         }
35
36         MovieBooking movieBooking = MovieBookingDTO.prepareEntity(movieBookingDTO);
37         movieBooking.setBookingAmount(calculateBookingAmount(movieBooking.getNoOfSeats(), movieBooking.getScreenName()));
38         Integer movieID = movieBookingRepository.save(movieBooking).getBookingId();
39         movieBookingDTO.setBookingId(movieID);
40
41
42         return MovieBookingDTO.prepareDTO(movieBooking);
43     }
44
45     @Override
46     public Double calculateBookingAmount(Integer noOfSeats, String screenName) {
```

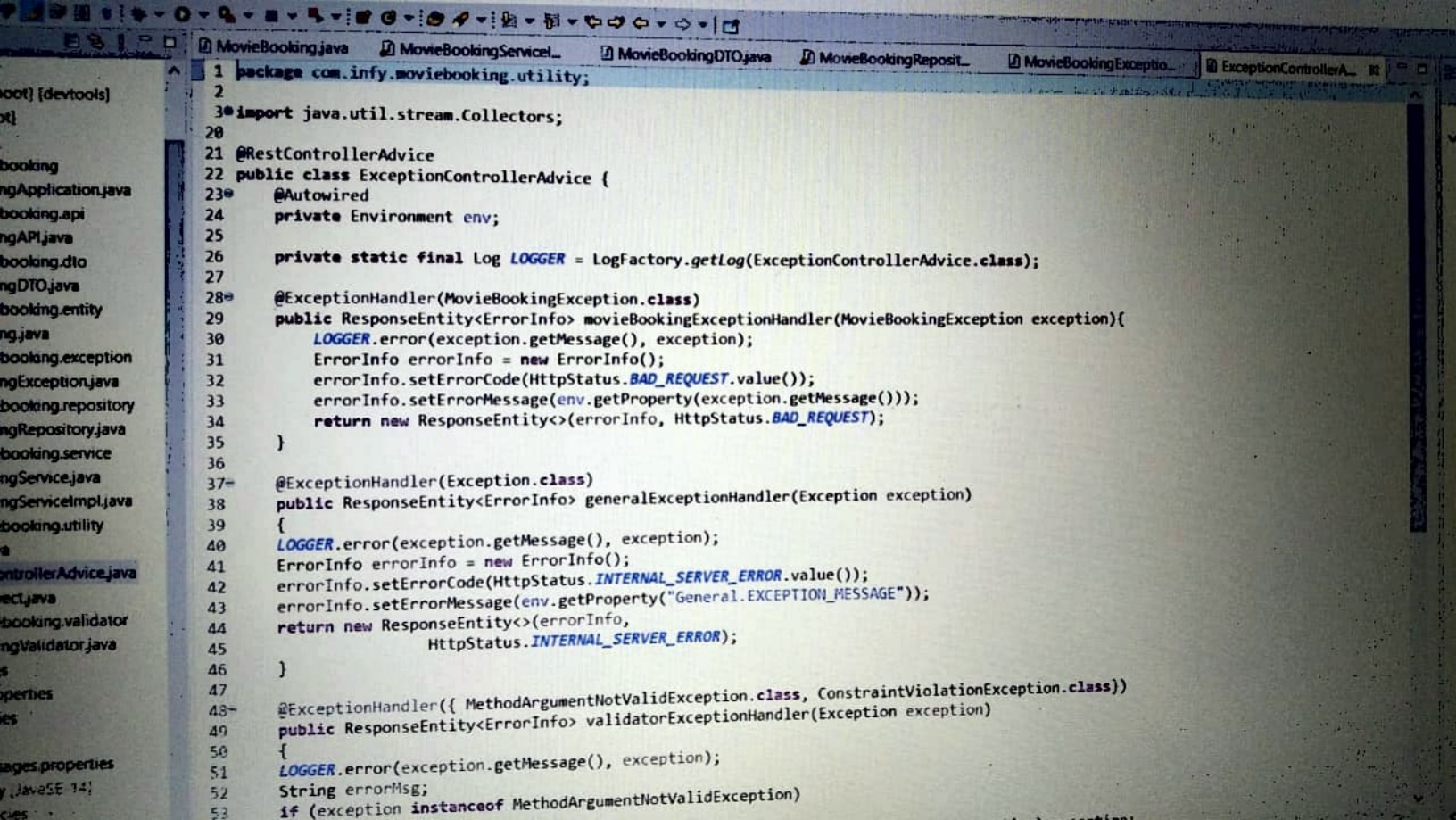
```
38 Integer movieID = movieBookingRepository.save(movieBooking).getBookingId();
39 movieBookingDTO.setBookingId(movieID);
40
41
42 return MovieBookingDTO.prepareDTO(movieBooking);
43 }
44
```

45 **@Override**

```
46 public Double calculateBookingAmount(Integer noOfSeats, String screenName) {
47     Double bookingAmount = 0.0;
48     if(screenName.equals("Rhombus")) {
49
50         bookingAmount=100.0 * noOfSeats;
51     }
52     else if (screenName.equals("Sapphire")) {
53         bookingAmount = 200.0* noOfSeats;
54     }
55     }
56     else {
57         bookingAmount = 300.0* noOfSeats;
58     }
59     return bookingAmount;
60 }
61
```

62 **@Override**

```
63 public List<MovieBookingDTO> getBookingByScreenName(String screenName) throws MovieBookingException {
64
65     List<MovieBooking> movieBookingList = movieBookingRepository.findByScreenName(screenName);
66     if(movieBookingList.isEmpty()) {
67         throw new MovieBookingException("MovieBookingService.NO_BOOKING");
68     }
69
70     List<MovieBookingDTO> dtoList = new ArrayList<>();
71     movieBookingList.stream().forEach(x->dtoList.add(new MovieBookingDTO().prepareDTO(x)));
72     return dtoList;
73 }
74
75 }
76
```

```
1 package com.infy.moviebooking.utility;
2
3 import java.util.stream.Collectors;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 @RestControllerAdvice
22 public class ExceptionControllerAdvice {
23     @Autowired
24     private Environment env;
25
26     private static final Log LOGGER = LoggerFactory.getLog(ExceptionControllerAdvice.class);
27
28     @ExceptionHandler(MovieBookingException.class)
29     public ResponseEntity<ErrorInfo> movieBookingExceptionHandler(MovieBookingException exception){
30         LOGGER.error(exception.getMessage(), exception);
31         ErrorInfo errorInfo = new ErrorInfo();
32         errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
33         errorInfo.setErrorMessage(env.getProperty(exception.getMessage()));
34         return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
35     }
36
37     @ExceptionHandler(Exception.class)
38     public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception)
39     {
40         LOGGER.error(exception.getMessage(), exception);
41         ErrorInfo errorInfo = new ErrorInfo();
42         errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
43         errorInfo.setErrorMessage(env.getProperty("General.EXCEPTION_MESSAGE"));
44         return new ResponseEntity<>(errorInfo,
45                                     HttpStatus.INTERNAL_SERVER_ERROR);
46     }
47
48     @ExceptionHandler({ MethodArgumentNotValidException.class, ConstraintViolationException.class})
49     public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception)
50     {
51         LOGGER.error(exception.getMessage(), exception);
52         String errorMsg;
53         if (exception instanceof MethodArgumentNotValidException)
```

```

40  LOGGER.error(exception.getMessage(), exception);
41  ErrorInfo errorInfo = new ErrorInfo();
42  errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
43  errorInfo.setErrorMessage(env.getProperty("General.EXCEPTION_MESSAGE"));
44  return new ResponseEntity<>(errorInfo,
45      HttpStatus.INTERNAL_SERVER_ERROR);
46  }
47
48  @ExceptionHandler({ MethodArgumentNotValidException.class, ConstraintViolationException.class })
49  public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception)
50  {
51      LOGGER.error(exception.getMessage(), exception);
52      String errorMsg;
53      if (exception instanceof MethodArgumentNotValidException)
54      {
55          MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
56          errorMsg = manvException.getBindingResult()
57              .getAllErrors()
58              .stream()
59              .map(ObjectError::getDefaultMessage)
60              .collect(Collectors.joining(", "));
61      }
62      else
63      {
64          ConstraintViolationException cvException = (ConstraintViolationException) exception;
65          errorMsg = cvException.getConstraintViolations()
66              .stream()
67              .map(ConstraintViolation::getMessage)
68              .collect(Collectors.joining(", "));
69      }
70
71      ErrorInfo errorInfo = new ErrorInfo();
72      errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
73      errorInfo.setErrorMessage(errorMsg);
74      return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
75  }
76  }
77  }

```


devtools]

ication.java

g.api

ava

g.dto

ava

entity

g.exception

tion.java

repository

itory.java

service

e.java

elmp1.java

utility

advice.java

validator

or.java

ities

14]

notations

es/test-annot

MovieBooking.java MovieBookingSe... MovieBookingDT... MovieBookingRe... MovieBookingEx... ExceptionContro... MovieBookingVal...

```
25 }
26
27 }
28
29 public static Boolean isValidPaymentType(String paymentType) {
30     String paymentType1 = "Card";
31     String paymentType2 = "Wallet";
32     String paymentType3 = "NetBanking";
33     if(paymentType1.equals(paymentType) || paymentType2.equals(paymentType) || paymentType3.equals(paymentType)) {
34         return true;
35     }
36     else
37         return false;
38
39 }
40
41
42
43 public static Boolean isValidCustomerPhoneNo(Long customerPhoneNo) {
44     String mob=customerPhoneNo.toString();
45     Boolean b=true;
46     if(mob.length()==10) {
47         char c=mob.charAt(0);
48         Integer a = Character.getNumericValue(c);
49         if((a>=3)&&(a<=9))
50
51             b=true;
52
53         else
54             b=false;
55         return b;
56
57     }
58     else
59         return false;
60 }
61
62 }
63
```

MovieBoo
Movie
valu
isVali
isVali

MovieBooking.java MovieBookingSe... MovieBookingDT... MovieBookingRe... MovieBookingEx... ExceptionContro... MovieBookingVal... 21

```
1 package com.infy.moviebooking.validator;
2
3 import java.time.LocalDate;
4
5
6
7
8
9
10
11 public class MovieBookingValidator {
12
13
14     private MovieBookingValidator(){
15
16
17     }
18
19     public static void validate(MovieBookingDTO movieBookingDTO) throws MovieBookingException{
20         if(Boolean.FALSE.equals(isValidCustomerPhoneNo(movieBookingDTO.getCustomerPhoneNo()))){
21             throw new MovieBookingException("Validator.INVALID_CUSTOMER_PHONE_NO");
22         }
23         if(Boolean.FALSE.equals(isValidPaymentType(movieBookingDTO.getPaymentType()))){
24             throw new MovieBookingException("Validator.INVALID_PAYMENT_TYPE");
25         }
26     }
27
28
29     public static Boolean isValidPaymentType(String paymentType) {
30         String paymentType1 = "Card";
31         String paymentType2 = "Wallet";
32         String paymentType3 = "NetBanking";
33         if(paymentType1.equals(paymentType) || paymentType2.equals(paymentType) || paymentType3.equals(paymentType)) {
34             return true;
35         }
36         else
37             return false;
38
39
40     }
41
42
43     public static Boolean isValidCustomerPhoneNo(Long customerPhoneNo) {
```



```
1 package com.infy.moviebooking.api;
2 import java.util.List;
18
19 @RestController
20 @RequestMapping(value="/api")
21 @Validated
22 public class MovieBookingAPI {
23
24     @Autowired
25     private MovieBookingService movieBookingService;
26
27     @PostMapping(value="/movie")
28     public ResponseEntity<MovieBookingDTO> bookMovie(@RequestBody @Valid MovieBookingDTO movieBookingDTO)
29         throws MovieBookingException{
30         MovieBookingDTO movieBooking = movieBookingService.bookMovie(movieBookingDTO);
31         return new ResponseEntity<>(movieBooking, HttpStatus.CREATED);
32     }
33
34     @GetMapping(value="/movie/{screenName}")
35     public ResponseEntity<List<MovieBookingDTO> > getBookingByScreenName(@PathVariable String screenName)
36         throws MovieBookingException{
37
38         List<MovieBookingDTO> mBDTO = movieBookingService.getBookingByScreenName(screenName);
39         return new ResponseEntity<>(mBDTO, HttpStatus.OK);
40     }
41
42 }
43
```