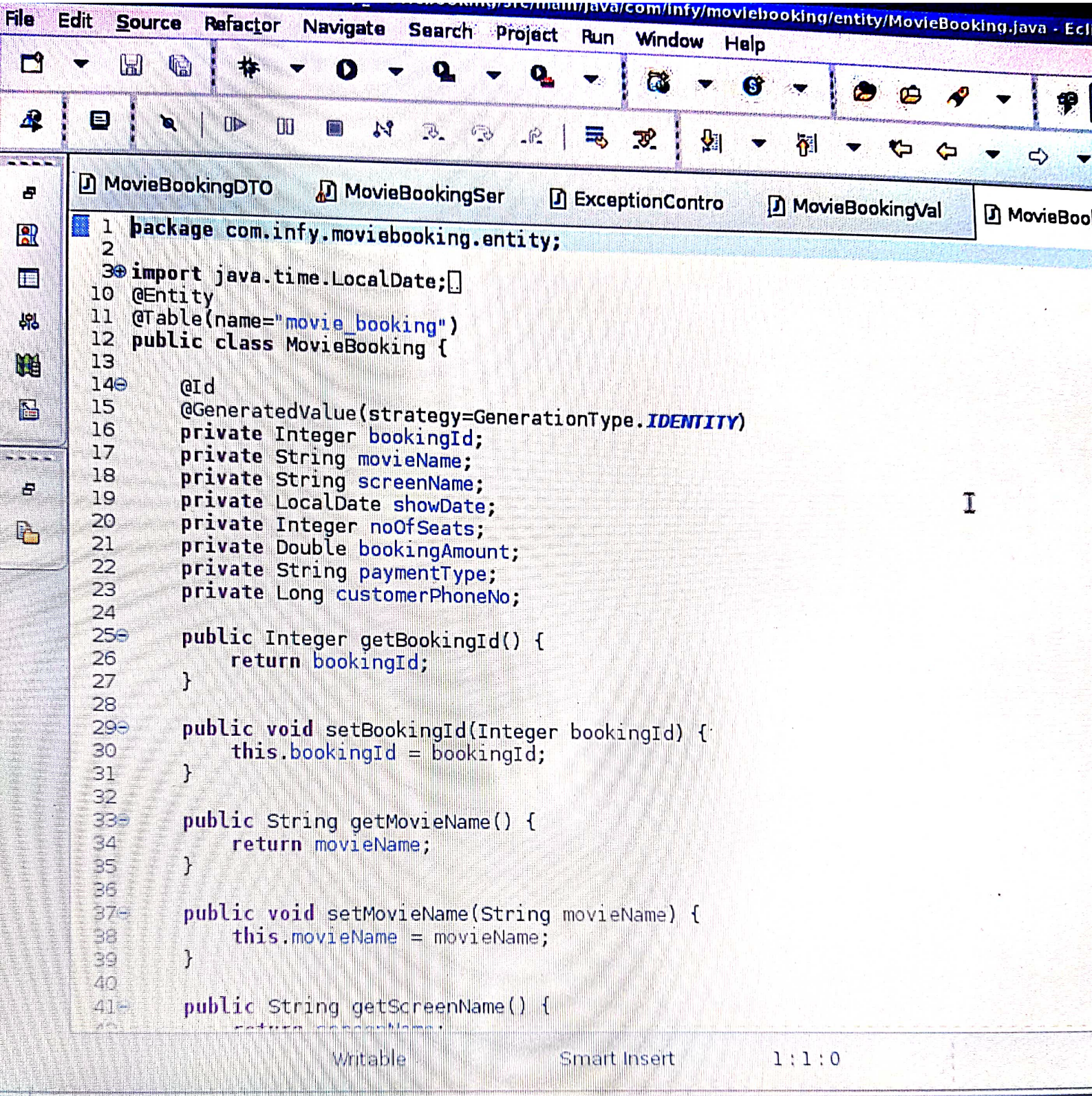
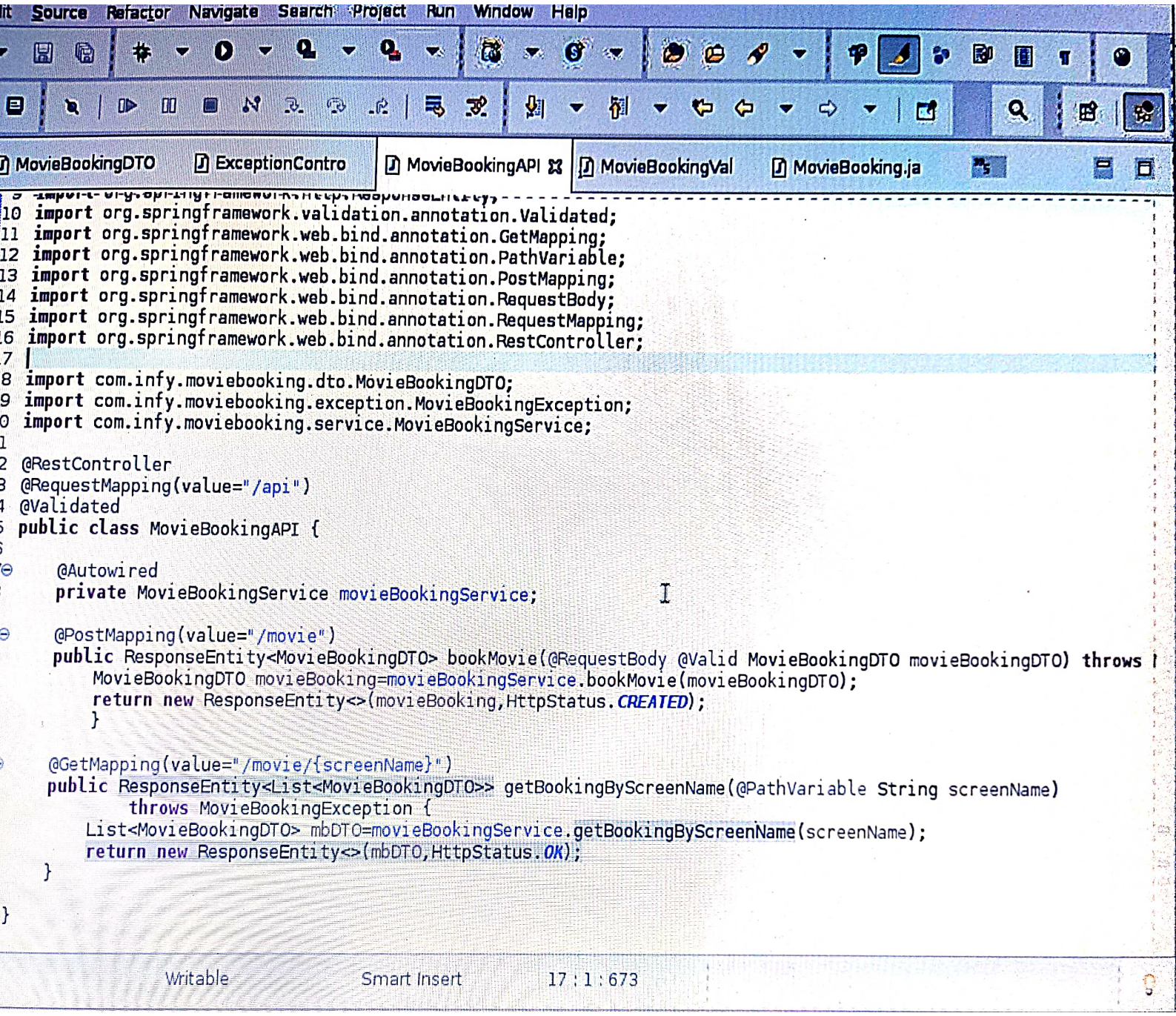
MovieBookingDTO | MovieBookingSer | ExceptionContro | MovieBookingVal | MovieBookingRep

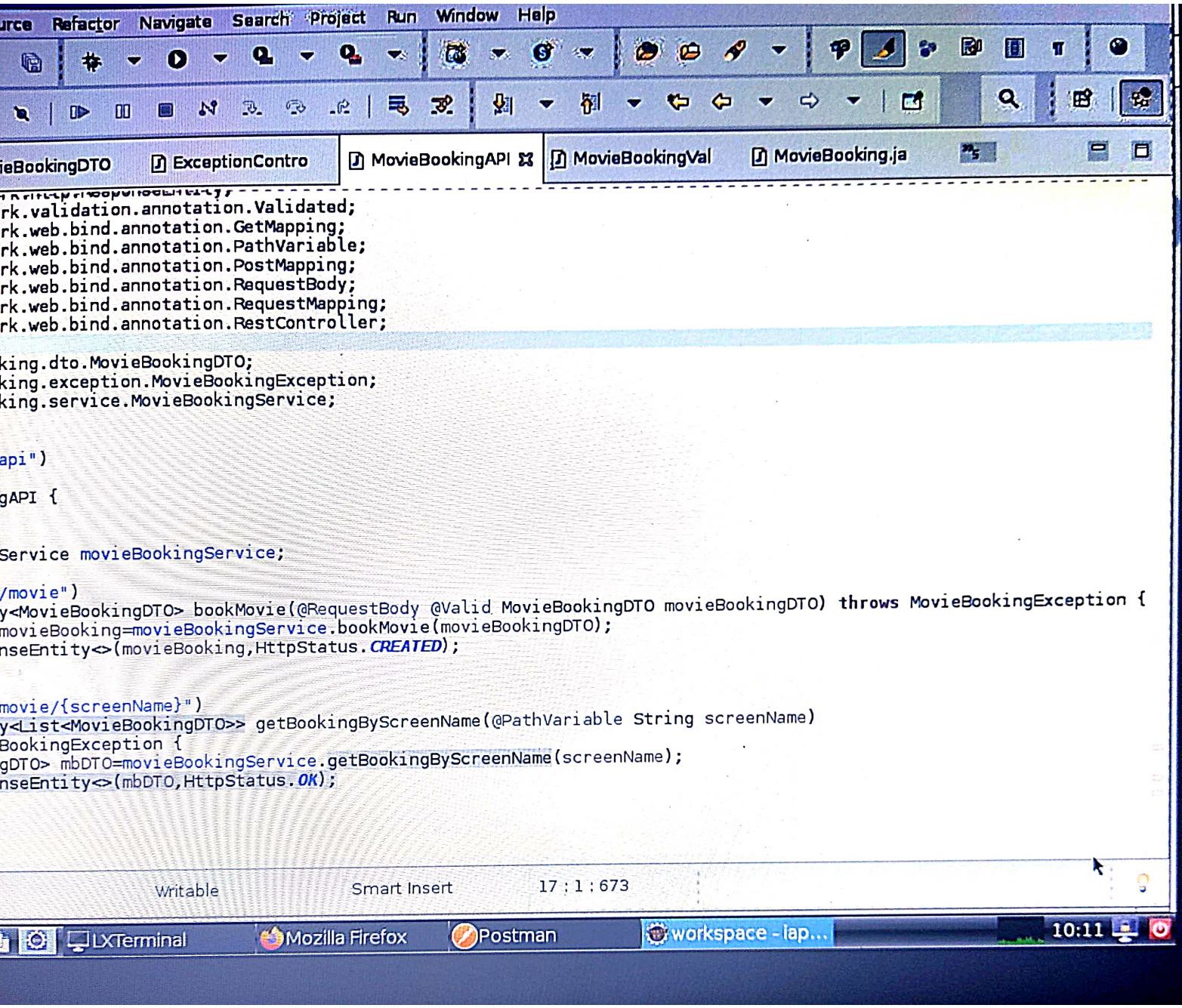
```
3  import java.time.LocalDate;
4
5  import javax.validation.constraints.FutureOrPresent;
6  import javax.validation.constraints.Max;
7  import javax.validation.constraints.Min;
8  import javax.validation.constraints.NotNull;
9  import javax.validation.constraints.Pattern;
10
11 import com.infy.moviebooking.entity.MovieBooking;
12
13 public class MovieBookingDTO {
14
15     private Integer bookingId;
16
17     @NotNull(message="{bookmovie.moviename.notpresent}")
18     private String movieName;
19
20     @NotNull(message="{bookmovie.screenname.notpresent}")
21     @Pattern(regexp="(Sapphire|Turquoise|Rhombus)",message="{bookmovie.screenname.invalid}")
22     private String screenName;
23
24     @NotNull(message="{bookmovie.showdate.notpresent}")
25     @FutureOrPresent(message="{bookmovie.showdate.invalid}")
26     private LocalDate showDate;
27
28     @NotNull(message="{bookmovie.noofseats.notpresent}")
29     @Min(value=1,message="{bookmovie.noofseats.invalid}")
30     @Max(value=5,message="{bookmovie.noofseats.invalid")
31     private Integer noOfSeats;
32
33     private Double bookingAmount;
34
35     private String paymentType;
36
37     private Long customerPhoneNo;
```

Writable | Smart Insert | 3 : 1 : 36

LXTerminal | Mozilla Firefox | Postman | workspace - iap...


File Edit Source Refactor Navigate Search Project Run Window Help
MovieBookingDTO
MovieBookingSer
ExceptionContro
MovieBookingVal
1 package com.infy.moviebooking.entity;
2
3 import java.time.LocalDate;
10 @Entity
11 @Table(name="movie_booking")
12 public class MovieBooking {
13
14 @Id
15 @GeneratedValue(strategy=GenerationType.IDENTITY)
16 private Integer bookingId;
17 private String movieName;
18 private String screenName;
19 private LocalDate showDate;
20 private Integer noOfSeats;
21 private Double bookingAmount;
22 private String paymentType;
23 private Long customerPhoneNo;
24
25 public Integer getBookingId() {
26 return bookingId;
27 }
28
29 public void setBookingId(Integer bookingId) {
30 this.bookingId = bookingId;
31 }
32
33 public String getMovieName() {
34 return movieName;
35 }
36
37 public void setMovieName(String movieName) {
38 this.movieName = movieName;
39 }
40
41 public String getScreenName() {
Writable
Smart Insert
1 : 1 : 0


Source Refactor Navigate Search Project Run Window Help
MovieBookingDTO
ExceptionContro
MovieBookingAPI
MovieBookingVal
MovieBooking.ja
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.infy.moviebooking.dto.MovieBookingDTO;
import com.infy.moviebooking.exception.MovieBookingException;
import com.infy.moviebooking.service.MovieBookingService;
@RestController
@RequestMapping(value="/api")
@Validated
public class MovieBookingAPI {
@Autowired
private MovieBookingService movieBookingService;
@PostMapping(value="/movie")
public ResponseEntity<MovieBookingDTO> bookMovie(@RequestBody @Valid MovieBookingDTO movieBookingDTO) throws
MovieBookingDTO movieBooking=movieBookingService.bookMovie(movieBookingDTO);
return new ResponseEntity<>(movieBooking,HttpStatus.CREATED);
}
@GetMapping(value="/movie/{screenName}")
public ResponseEntity<List<MovieBookingDTO>> getBookingByScreenName(@PathVariable String screenName)
throws MovieBookingException {
List<MovieBookingDTO> mbDTO=movieBookingService.getBookingByScreenName(screenName);
return new ResponseEntity<>(mbDTO,HttpStatus.OK);
}
}
Writable
Smart Insert
17 : 1 : 673

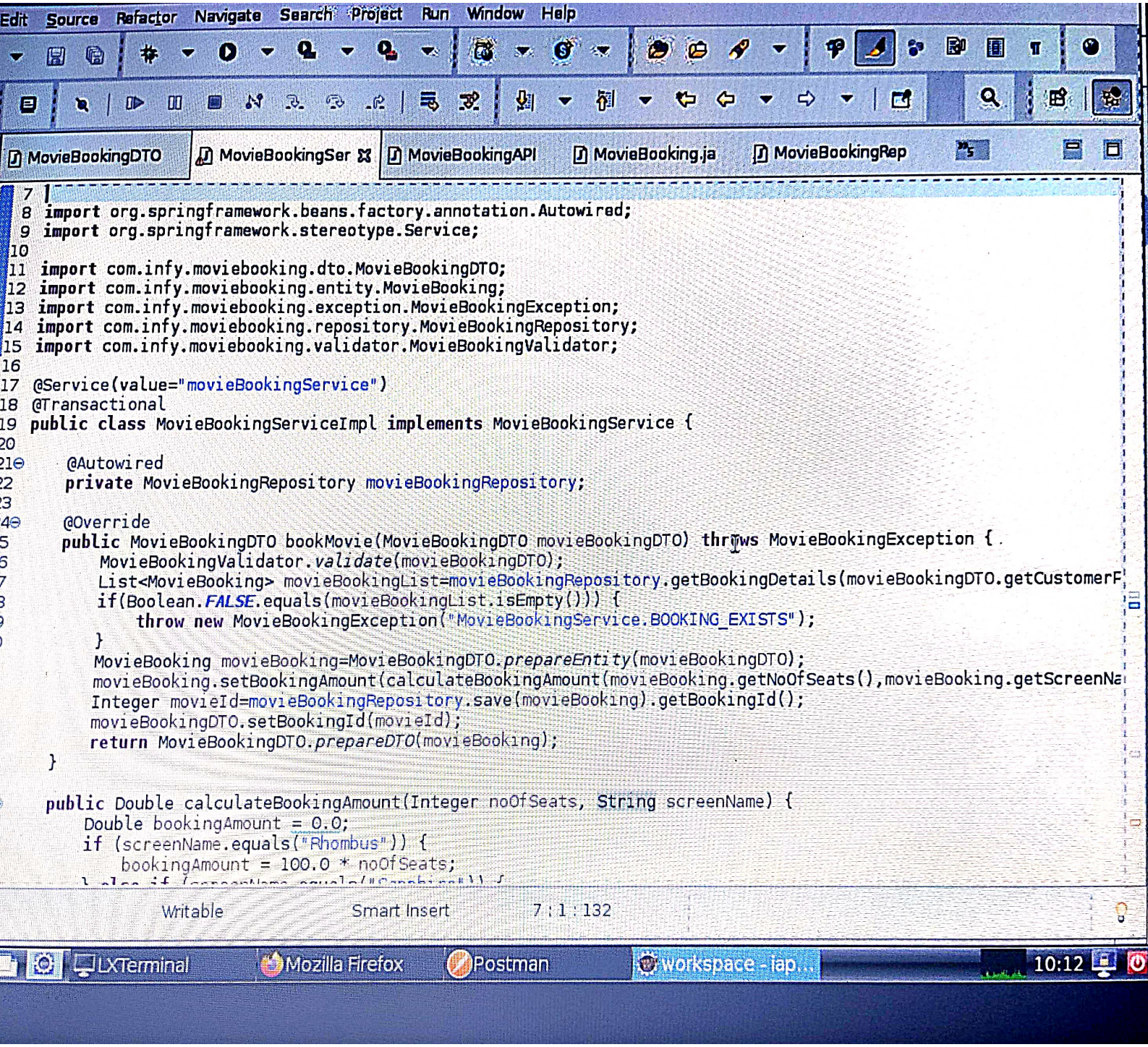

urce Refactor Navigate Search Project Run Window Help
ieBookingDTO
ExceptionContro
MovieBookingAPI
MovieBookingVal
MovieBooking.ja
rk.validation.annotation.Validated;
rk.web.bind.annotation.GetMapping;
rk.web.bind.annotation.PathVariable;
rk.web.bind.annotation.PostMapping;
rk.web.bind.annotation.RequestBody;
rk.web.bind.annotation.RequestMapping;
rk.web.bind.annotation.RestController;
king.dto.MovieBookingDTO;
king.exception.MovieBookingException;
king.service.MovieBookingService;
api")
gAPI {
Service movieBookingService;
/movie")
y<MovieBookingDTO> bookMovie(@RequestBody @Valid MovieBookingDTO movieBookingDTO) throws MovieBookingException {
movieBooking=movieBookingService.bookMovie(movieBookingDTO);
nseEntity<>(movieBooking,HttpStatus.CREATED);
movie/{screenName}")
y<List<MovieBookingDTO>> getBookingByScreenName(@PathVariable String screenName)
BookingException {
gDTO> mbDTO=movieBookingService.getBookingByScreenName(screenName);
nseEntity<>(mbDTO,HttpStatus.OK);
Writable
Smart Insert
17 : 1 : 673
LXTerminal
Mozilla Firefox
Postman
workspace - lap...
10:11

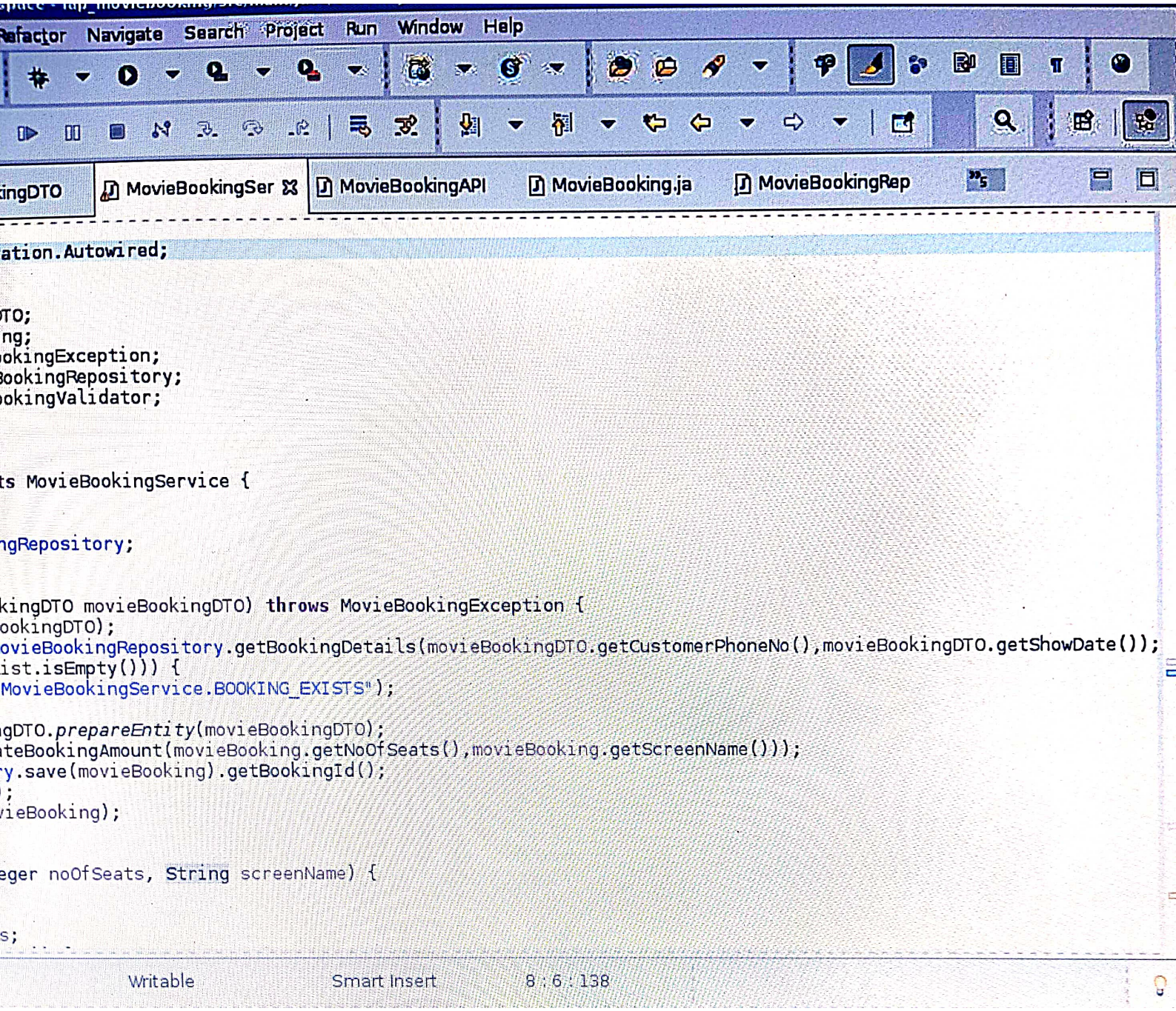
Edit Source Refactor Navigate Search Project Run Window Help

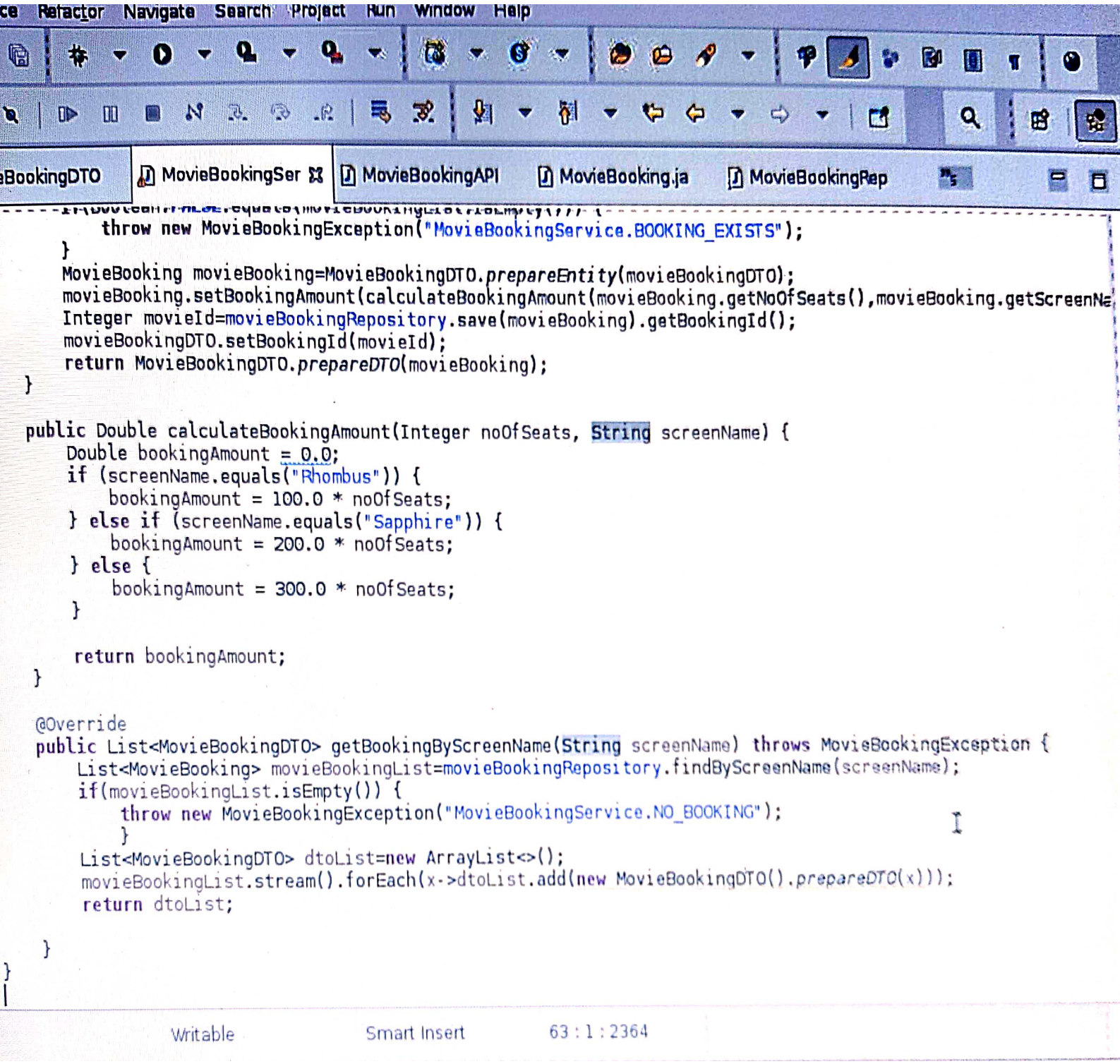
MovieBookingDTO | ExceptionContro | MovieBookingAPI | MovieBooking.ja | MovieBookingRep

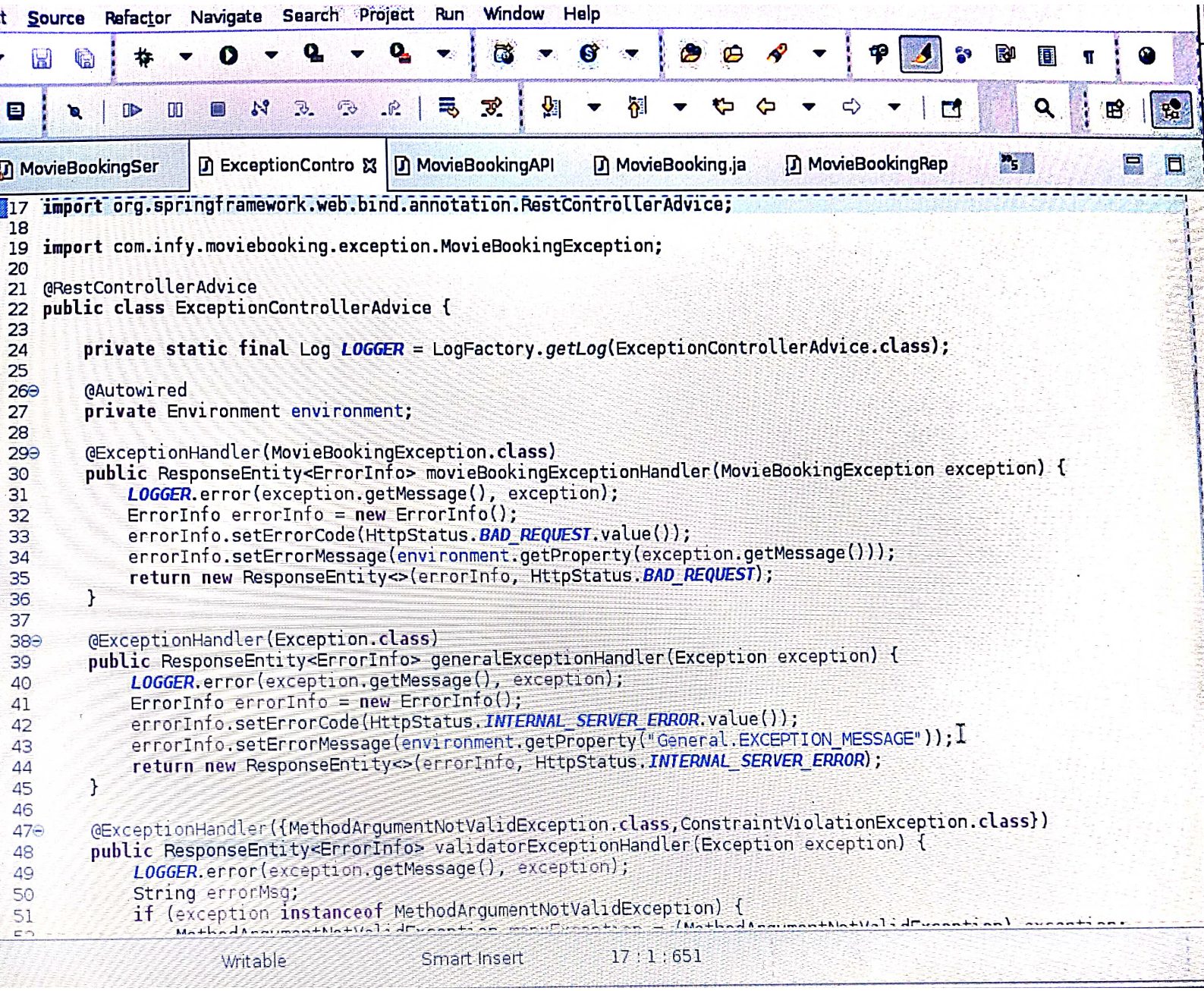
```
1  package com.infy.moviebooking.repository;
2
3  import java.time.LocalDate;
10
11 public interface MovieBookingRepository extends CrudRepository<MovieBooking,Integer> {
12     @Query("select m from MovieBooking m where m.customerPhoneNo=?1 and m.showDate=?2")
13     List<MovieBooking> getBookingDetails(Long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

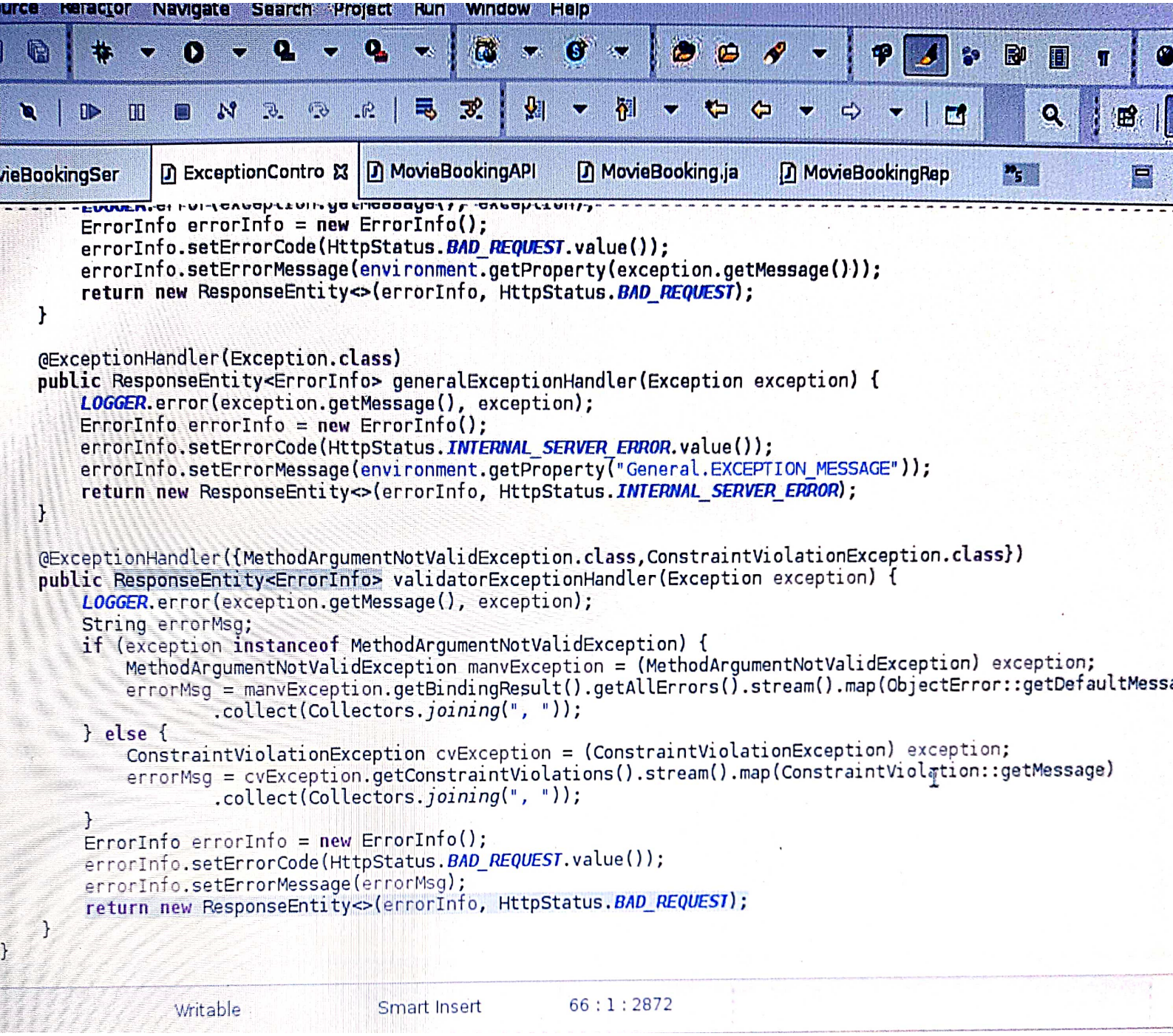
Writable Smart Insert 1 : 1 : 0

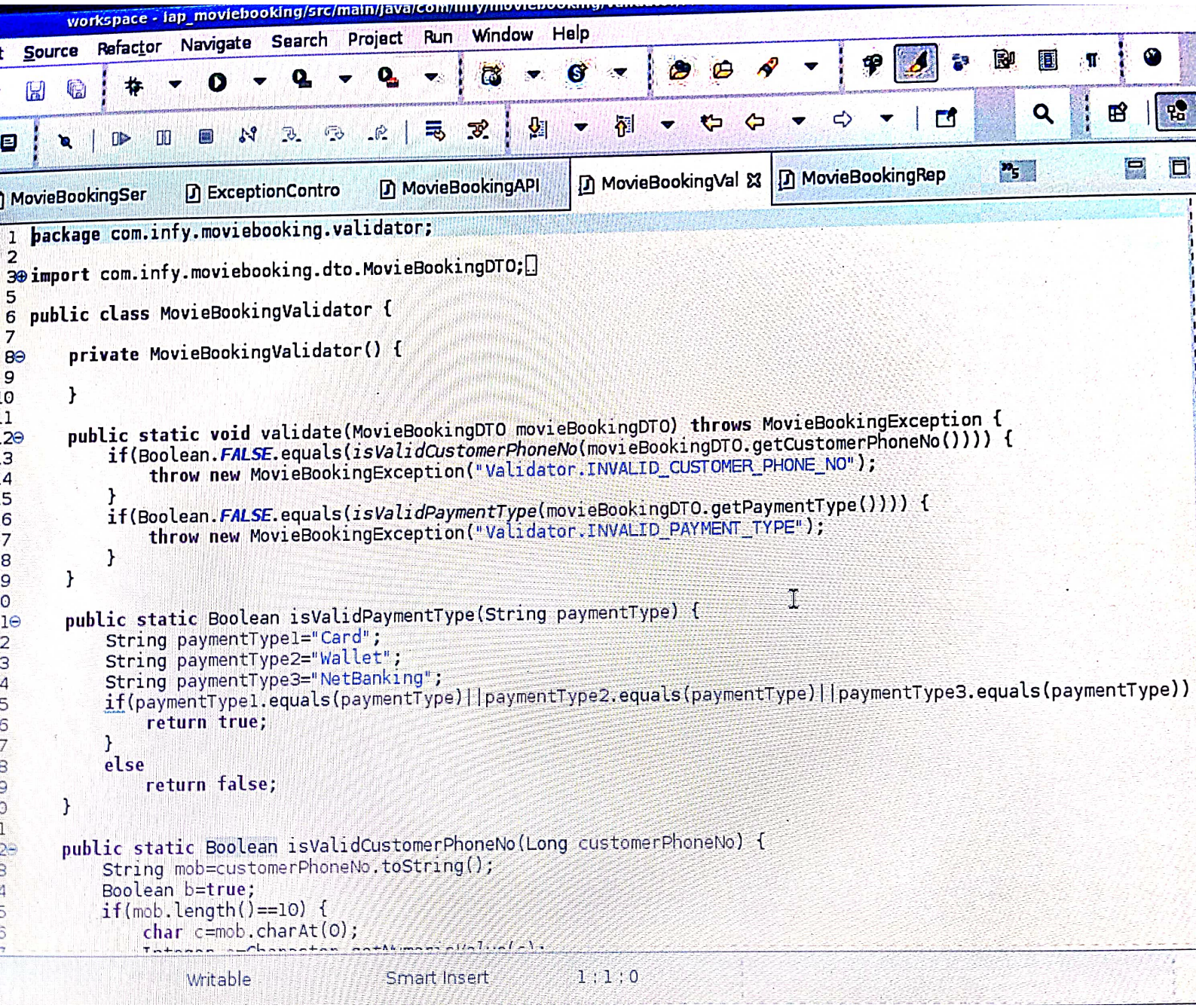

Edit Source Refactor Navigate Search Project Run Window Help
MovieBookingDTO
MovieBookingSer
MovieBookingAPI
MovieBooking.ja
MovieBookingRep
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.infy.moviebooking.dto.MovieBookingDTO;
import com.infy.moviebooking.entity.MovieBooking;
import com.infy.moviebooking.exception.MovieBookingException;
import com.infy.moviebooking.repository.MovieBookingRepository;
import com.infy.moviebooking.validator.MovieBookingValidator;
@Service(value="movieBookingService")
@Transactional
public class MovieBookingServiceImpl implements MovieBookingService {
@Autowired
private MovieBookingRepository movieBookingRepository;
@Override
public MovieBookingDTO bookMovie(MovieBookingDTO movieBookingDTO) throws MovieBookingException {
MovieBookingValidator.validate(movieBookingDTO);
List<MovieBooking> movieBookingList=movieBookingRepository.getBookingDetails(movieBookingDTO.getCustomerF
if(Boolean.FALSE.equals(movieBookingList.isEmpty())) {
throw new MovieBookingException("MovieBookingService.BOOKING_EXISTS");
}
MovieBooking movieBooking=MovieBookingDTO.prepareEntity(movieBookingDTO);
movieBooking.setBookingAmount(calculateBookingAmount(movieBooking.getNoOfSeats(),movieBooking.getScreenNa
Integer movieId=movieBookingRepository.save(movieBooking).getBookingId();
movieBookingDTO.setBookingId(movieId);
return MovieBookingDTO.prepareDTO(movieBooking);
}
public Double calculateBookingAmount(Integer noOfSeats, String screenName) {
Double bookingAmount = 0.0;
if (screenName.equals("Rhombus")) {
bookingAmount = 100.0 * noOfSeats;
Writable
Smart Insert
7 : 1 : 132
LXTerminal
Mozilla Firefox
Postman
workspace - iap...
10:12

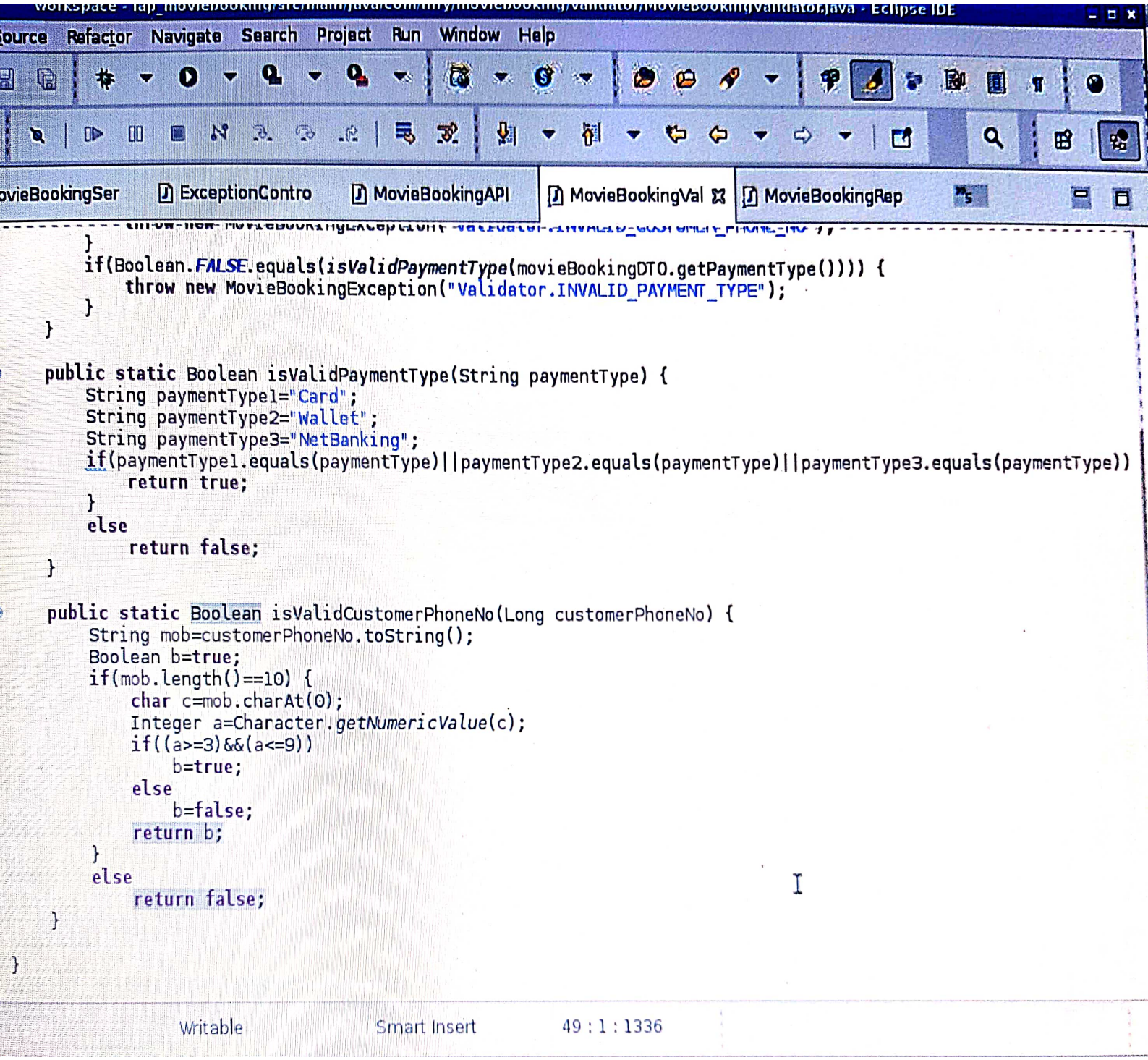

Refactor Navigate Search Project Run Window Help
kingDTO
MovieBookingSer
MovieBookingAPI
MovieBooking.ja
MovieBookingRep
ation.Autowired;
TO;
ng;
ookingException;
BookingRepository;
ookingValidator;
ts MovieBookingService {
ngRepository;
kingDTO movieBookingDTO) throws MovieBookingException {
ookingDTO);
ovieBookingRepository.getBookingDetails(movieBookingDTO.getCustomerPhoneNo(),movieBookingDTO.getShowDate());
ist.isEmpty())) {
MovieBookingService.BOOKING_EXISTS");
ngDTO.prepareEntity(movieBookingDTO);
teBookingAmount(movieBooking.getNoOfSeats(),movieBooking.getScreenName()));
y.save(movieBooking).getBookingId();
;
ieBooking);
eger noOfSeats, String screenName) {
s;
Writable
Smart Insert
8 : 6 : 138


ce Refactor Navigate Search Project Run Window Help
eBookingDTO
MovieBookingSer
MovieBookingAPI
MovieBooking.ja
MovieBookingRep
throw new MovieBookingException("MovieBookingService.BOOKING_EXISTS");
}
MovieBooking movieBooking=MovieBookingDTO.prepareEntity(movieBookingDTO);
movieBooking.setBookingAmount(calculateBookingAmount(movieBooking.getNoOfSeats(),movieBooking.getScreenNa
Integer movieId=movieBookingRepository.save(movieBooking).getBookingId();
movieBookingDTO.setBookingId(movieId);
return MovieBookingDTO.prepareDTO(movieBooking);
}
public Double calculateBookingAmount(Integer noOfSeats, String screenName) {
Double bookingAmount = 0.0;
if (screenName.equals("Rhombus")) {
bookingAmount = 100.0 * noOfSeats;
} else if (screenName.equals("Sapphire")) {
bookingAmount = 200.0 * noOfSeats;
} else {
bookingAmount = 300.0 * noOfSeats;
}
return bookingAmount;
}
@Override
public List<MovieBookingDTO> getBookingByScreenName(String screenName) throws MovieBookingException {
List<MovieBooking> movieBookingList=movieBookingRepository.findByScreenName(screenName);
if(movieBookingList.isEmpty()) {
throw new MovieBookingException("MovieBookingService.NO_BOOKING");
}
List<MovieBookingDTO> dtoList=new ArrayList<>();
movieBookingList.stream().forEach(x->dtoList.add(new MovieBookingDTO().prepareDTO(x)));
return dtoList;
}
}
Writable
Smart Insert
63 : 1 : 2364


Source Refactor Navigate Search Project Run Window Help
MovieBookingSer
ExceptionContro
MovieBookingAPI
MovieBooking.ja
MovieBookingRep
17 import org.springframework.web.bind.annotation.RestControllerAdvice;
18
19 import com.infy.moviebooking.exception.MovieBookingException;
20
21 @RestControllerAdvice
22 public class ExceptionControllerAdvice {
23
24 private static final Log LOGGER = LogFactory.getLog(ExceptionControllerAdvice.class);
25
26 @Autowired
27 private Environment environment;
28
29 @ExceptionHandler(MovieBookingException.class)
30 public ResponseEntity<ErrorInfo> movieBookingExceptionHandler(MovieBookingException exception) {
31 LOGGER.error(exception.getMessage(), exception);
32 ErrorInfo errorInfo = new ErrorInfo();
33 errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
34 errorInfo.setErrorMessage(environment.getProperty(exception.getMessage()));
35 return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
36 }
37
38 @ExceptionHandler(Exception.class)
39 public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception) {
40 LOGGER.error(exception.getMessage(), exception);
41 ErrorInfo errorInfo = new ErrorInfo();
42 errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
43 errorInfo.setErrorMessage(environment.getProperty("General.EXCEPTION_MESSAGE"));
44 return new ResponseEntity<>(errorInfo, HttpStatus.INTERNAL_SERVER_ERROR);
45 }
46
47 @ExceptionHandler({MethodArgumentNotValidException.class,ConstraintViolationException.class})
48 public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception) {
49 LOGGER.error(exception.getMessage(), exception);
50 String errorMsg;
51 if (exception instanceof MethodArgumentNotValidException) {
Writable
Smart Insert
17 : 1 : 651


urce Refactor Navigate Search Project Run Window Help
vieBookingSer
ExceptionContro
MovieBookingAPI
MovieBooking.ja
MovieBookingRep
ErrorInfo errorInfo = new ErrorInfo();
errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
errorInfo.setErrorMessage(environment.getProperty(exception.getMessage()));
return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
}
@ExceptionHandler(Exception.class)
public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception) {
LOGGER.error(exception.getMessage(), exception);
ErrorInfo errorInfo = new ErrorInfo();
errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
errorInfo.setErrorMessage(environment.getProperty("General.EXCEPTION_MESSAGE"));
return new ResponseEntity<>(errorInfo, HttpStatus.INTERNAL_SERVER_ERROR);
}
@ExceptionHandler({MethodArgumentNotValidException.class,ConstraintViolationException.class})
public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception) {
LOGGER.error(exception.getMessage(), exception);
String errorMsg;
if (exception instanceof MethodArgumentNotValidException) {
MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
errorMsg = manvException.getBindingResult().getAllErrors().stream().map(ObjectError::getDefaultMessa
.collect(Collectors.joining(", "));
} else {
ConstraintViolationException cvException = (ConstraintViolationException) exception;
errorMsg = cvException.getConstraintViolations().stream().map(ConstraintViolation::getMessage)
.collect(Collectors.joining(", "));
}
ErrorInfo errorInfo = new ErrorInfo();
errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
errorInfo.setErrorMessage(errorMsg);
return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
}
}
Writable
Smart Insert
66 : 1 : 2872


workspace - iap_moviebooking/src/main/java/com/infy/moviebooking/validator
Source Refactor Navigate Search Project Run Window Help
MovieBookingSer
ExceptionContro
MovieBookingAPI
MovieBookingVal
MovieBookingRep
1 package com.infy.moviebooking.validator;
2
3 import com.infy.moviebooking.dto.MovieBookingDTO;
5
6 public class MovieBookingValidator {
7
8 private MovieBookingValidator() {
9
10 }
11
12 public static void validate(MovieBookingDTO movieBookingDTO) throws MovieBookingException {
13 if(Boolean.FALSE.equals(isValidCustomerPhoneNo(movieBookingDTO.getCustomerPhoneNo()))) {
14 throw new MovieBookingException("Validator.INVALID_CUSTOMER_PHONE_NO");
15 }
16 if(Boolean.FALSE.equals(isValidPaymentType(movieBookingDTO.getPaymentType()))) {
17 throw new MovieBookingException("Validator.INVALID_PAYMENT_TYPE");
18 }
19 }
20
21 public static Boolean isValidPaymentType(String paymentType) {
22 String paymentType1="Card";
23 String paymentType2="Wallet";
24 String paymentType3="NetBanking";
25 if(paymentType1.equals(paymentType)||paymentType2.equals(paymentType)||paymentType3.equals(paymentType))
26 return true;
27 }
28 else
29 return false;
30 }
31
32 public static Boolean isValidCustomerPhoneNo(Long customerPhoneNo) {
33 String mob=customerPhoneNo.toString();
34 Boolean b=true;
35 if(mob.length()==10) {
36 char c=mob.charAt(0);
Writable
Smart Insert
1 : 1 : 0


Source Refactor Navigate Search Project Run Window Help
MovieBookingSer
ExceptionContro
MovieBookingAPI
MovieBookingVal
MovieBookingRep
}
if(Boolean.FALSE.equals(isValidPaymentType(movieBookingDTO.getPaymentType()))) {
throw new MovieBookingException("Validator.INVALID_PAYMENT_TYPE");
}
}
public static Boolean isValidPaymentType(String paymentType) {
String paymentType1="Card";
String paymentType2="Wallet";
String paymentType3="NetBanking";
if(paymentType1.equals(paymentType)||paymentType2.equals(paymentType)||paymentType3.equals(paymentType))
return true;
}
else
return false;
}
public static Boolean isValidCustomerPhoneNo(Long customerPhoneNo) {
String mob=customerPhoneNo.toString();
Boolean b=true;
if(mob.length()==10) {
char c=mob.charAt(0);
Integer a=Character.getNumericValue(c);
if((a>=3)&&(a<=9))
b=true;
else
b=false;
return b;
}
else
return false;
}
}
Writable
Smart Insert
49 : 1 : 1336

BeanValidation	7	1	8
ExceptionControllerAdvice	4	1	5
MovieBooking	2	0	2
MovieBookingAPI - Annotations	3	0	3
MovieBookingAPI - bookMovie	2	0	2
MovieBookingAPI - getBookingByScreenName	2	0	2
MovieBookingDTO	5	1	6
MovieBookingServiceImpl - Annotations	3	0	3
MovieBookingServiceImpl - bookMovie	4	0	4
MovieBookingServiceImpl - getBookingByScreenName	2	0	2
MovieBookingValidator - isValidCustomerPhoneNo	8	0	8
MovieBookingValidator - isValidPaymentType	8	0	8
MovieBookingValidator - validate	2	0	2
Total	52	3	55