Creating entity class;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@NotNull

@Size(min = 2, max = 100)

private String title;

@NotNull

@Size(min = 2, max = 50)

private String author;

@Min(1)

private int price;

@Version

private int version;

// Getters and setters

}

@Entity

public class Customer {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@NotNull

@Size(min = 2, max = 50)

private String name;

@NotNull

@Size(min = 5, max = 100)

private String email;

@Version

private int version;

// Getters and setters

}

//Repository layer

public interface BookRepository extends JpaRepository<Book, Long> {}

public interface CustomerRepository extends JpaRepository<Customer, Long> {}

//Service Layer

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

public Book createBook(Book book) {

return bookRepository.save(book);

}

public Book getBook(Long id) {

return bookRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Book not found"));

}

public Book updateBook(Long id, Book bookDetails) {

Book book = getBook(id);

book.setTitle(bookDetails.getTitle());

book.setAuthor(bookDetails.getAuthor());

book.setPrice(bookDetails.getPrice());

return bookRepository.save(book);

}

public void deleteBook(Long id) {

Book book = getBook(id);

bookRepository.delete(book);

}

}

//Controller Layer

@RestController

@RequestMapping("/api/books")

public class BookController {

@Autowired

private BookService bookService;

@PostMapping

public ResponseEntity<Book> createBook(@Valid @RequestBody Book book) {

Book createdBook = bookService.createBook(book);

return ResponseEntity.status(HttpStatus.CREATED).body(createdBook);

}

@GetMapping("/{id}")

public ResponseEntity<Book> getBook(@PathVariable Long id) {

Book book = bookService.getBook(id);

return ResponseEntity.ok(book);

}

@PutMapping("/{id}")

public ResponseEntity<Book> updateBook(@PathVariable Long id, @Valid @RequestBody Book bookDetails) {

Book updatedBook = bookService.updateBook(id, bookDetails);

return ResponseEntity.ok(updatedBook);

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteBook(@PathVariable Long id) {

bookService.deleteBook(id);

return ResponseEntity.noContent().build();

}

}

//Exceptional Handling

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(ResourceNotFoundException.class)

public ResponseEntity<?> resourceNotFoundException(ResourceNotFoundException ex, WebRequest request) {

return new ResponseEntity<>(new ErrorDetails(new Date(), ex.getMessage(), request.getDescription(false)), HttpStatus.NOT\_FOUND);

}

@ExceptionHandler(OptimisticLockingFailureException.class)

public ResponseEntity<?> handleOptimisticLockingFailure(OptimisticLockingFailureException ex, WebRequest request) {

return new ResponseEntity<>(new ErrorDetails(new Date(), "Concurrent update detected", request.getDescription(false)), HttpStatus.CONFLICT);

}

// Add more handlers as needed

}