src\main\webapp\WEB-INF\views\addBook.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link href="<c:url value='/resources/css/bootstrap.css'/>" rel="stylesheet">
<title><spring:message code="addBook.form.title.label" /></title>
</head>
<body>
    <nav class="navbar navbar-expand navbar-dark bg-dark">
        <div class="container">
            <div class="navbar-header">
                <a class="navbar-brand" href="./home">Home</a>
            </div>
        </div>
    </nav>

    <div class="jumbotron">
        <div class="container">
            <h1 class="display-3">
                <spring:message code="addBook.form.title.label" />
            </h1>
        </div>
    </div>
    <!-- 로그아웃 버튼 만들기 -->
    <div class="container">
        <div class="float-right">
            <form:form action="${pageContext.request.contextPath}/logout"
                method="POST">
                <input type="submit" class="btn btn-sm btn-success" value="Logout" />
            </form:form>

            <div class="float-right" style="padding-right: 30px">
                <a href="?language=ko">Korean</a> | <a href="?language=en">English</a>
            </div>
        </div>
    </div>

    <form:form modelAttribute="NewBook" class="form-horizontal"
        action="./add" enctype="multipart/form-data">
        <fieldset>
            <legend>
                <spring:message code="addBook.form.subtitle.label" />
            </legend>
            <div class="form-group row">
                <label class="col-sm-2 control-label">
                    <spring:message code="addBook.form.bookId.label" />
```

```
52                  </label>
53                  <div class="col-sm-3">
54                      <form:input path="bookId" class="form-control" />
55                  </div>
56              </div>
57              <div class="form-group row">
58                  <label class="col-sm-2 control-label"><spring:message
59                          code="addBook.form.name.label" /></label>
60                  <div class="col-sm-3">
61                      <form:input path="name" class="form-control" />
62                  </div>
63              </div>
64              <div class="form-group row">
65                  <label class="col-sm-2 control-label">
66                      <spring:message code="addBook.form.unitPrice.label" />
67                  </label>
68                  <div class="col-sm-3">
69                      <form:input path="unitPrice" class="form-control" />
70                  </div>
71              </div>
72              <div class="form-group row">
73                  <label class="col-sm-2 control-label">
74                      <spring:message code="addBook.form.author.label" />
75                  </label>
76                  <div class="col-sm-3">
77                      <form:input path="author" class="form-control" />
78                  </div>
79              </div>
80              <div class="form-group row">
81                  <label class="col-sm-2 control-label">
82                      <spring:message code="addBook.form.description.label" />
83                  </label>
84                  <div class="col-sm-5">
85                      <form:textarea path="description" cols="50" rows="2"
86                          class="form-control" />
87                  </div>
88              </div>
89              <div class="form-group row">
90                  <label class="col-sm-2 control-label">
91                      <spring:message code="addBook.form.publisher.label" />
92                  </label>
93                  <div class="col-sm-3">
94                      <form:input path="publisher" class="form-control" />
95                  </div>
96              </div>
97              <div class="form-group row">
98                  <label class="col-sm-2 control-label"><spring:message
99                          code="addBook.form.category.label" /></label>
100                 <div class="col-sm-3">
101                     <form:input path="category" class="form-control" />
102                 </div>
103             </div>
104             <div class="form-group row">
105                 <label class="col-sm-2 control-label">
```

```
106                     <spring:message code="addBook.form.unitsInStock.label" />
107                 </label>
108                 <div class="col-sm-3">
109                     <form:input path="unitsInStock" class="form-control" />
110                 </div>
111             </div>
112             <div class="form-group row">
113                 <label class="col-sm-2 control-label">
114                     <spring:message code="addBook.form.releaseDate.label" />
115                 </label>
116                 <div class="col-sm-3">
117                     <form:input path="releaseDate" class="form-control" />
118                 </div>
119             </div>
120             <div class="form-group row">
121                 <label class="col-sm-2 control-label">
122                     <spring:message code="addBook.form.condition.label" />
123                 </label>
124                 <div class="col-sm-3">
125                     <form:radiobutton path="condition" value="New" /> New
126                     <form:radiobutton path="condition" value="Old" /> Old
127                     <form:radiobutton path="condition" value="E-Book" /> E-Book
128                 </div>
129             </div>
130             <div class="form-group row">
131                 <label class="col-sm-2 control-label">
132                     <spring:message code="addBook.form.bookImage.label" /></label>
133                 <div class="col-sm-7">
134                     <form:input path="bookImage" type="file" class="form-control" />
135                 </div>
136             </div>
137             <div class="form-group row">
138                 <div class="col-sm-offset-2 col-sm-10">
139                     <input type="submit" class="btn btn-primary"
140                            value="<spring:message code='addBook.form.button.label' />" />
141                 </div>
142             </div>
143         </fieldset>
144     </form:form>
145
146     <hr>
147     <footer>
148         <p>&copy; BookMarket</p>
149     </footer>
150 </body>
151 </html>
```

src\main\java\com\springmvc\domain\Book.java

```java
package com.springmvc.domain;

import java.util.Objects;

import org.springframework.web.multipart.MultipartFile;

public class Book { // DB에 저장되는 객체
    private String bookId;
    private String name;
    private int unitPrice;
    private String author;
    private String description;
    private String publisher;
    private String category;
    private long unitsInStock;
    private String releaseDate;
    private String condition;

    private String fileName;
    private MultipartFile bookImage;

    public Book() { //기본생성자
    }

    public Book(String bookId, String name, int unitPrice) {
        this.bookId = bookId;
        this.name = name;
        this.unitPrice = unitPrice;
    }

    public String getBookId() {
        return bookId;
    }

    public void setBookId(String bookId) {
        this.bookId = bookId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getUnitPrice() {
        return unitPrice;
    }

    public void setUnitPrice(int unitPrice) {
```

```
52          this.unitPrice = unitPrice;
53      }
54
55      public String getAuthor() {
56          return author;
57      }
58
59      public void setAuthor(String author) {
60          this.author = author;
61      }
62
63      public String getDescription() {
64          return description;
65      }
66
67      public void setDescription(String description) {
68          this.description = description;
69      }
70
71      public String getPublisher() {
72          return publisher;
73      }
74
75      public void setPublisher(String publisher) {
76          this.publisher = publisher;
77      }
78
79      public String getCategory() {
80          return category;
81      }
82
83      public void setCategory(String category) {
84          this.category = category;
85      }
86
87      public long getUnitsInStock() {
88          return unitsInStock;
89      }
90
91      public void setUnitsInStock(long unitsInStock) {
92          this.unitsInStock = unitsInStock;
93      }
94
95      public String getReleaseDate() {
96          return releaseDate;
97      }
98
99      public void setReleaseDate(String releaseDate) {
100         this.releaseDate = releaseDate;
101     }
102
103     public String getCondition() {
104         return condition;
105     }
```

```
106
107     public void setCondition(String condition) {
108         this.condition = condition;
109     }
110
111     public MultipartFile getBookImage() {
112         return bookImage;
113     }
114
115     public void setBookImage(MultipartFile bookImage) {
116         this.bookImage = bookImage;
117     }
118
119     public String getFileName() {
120         return fileName;
121     }
122
123     public void setFileName(String fileName) {
124         this.fileName = fileName;
125     }
126
127     // hashCode, equals: 인스턴스는 다르지만 같은 객체라는걸 판별 시켜주는 메소드
128     // 둘이 일치하면 같은 객체임
129     @Override
130     public int hashCode() {
131         return Objects.hash(author, bookId, bookImage, category, condition, description,
    fileName, name, publisher,
132                 releaseDate, unitPrice, unitsInStock);
133     }
134
135     @Override
136     public boolean equals(Object obj) {
137         if (this == obj) // jvm heap안의 메모리에 있는 주소를 가르킴
138             return true;
139         if (obj == null)
140             return false;
141         if (getClass() != obj.getClass())
142             return false;
143         Book other = (Book) obj;
144         return Objects.equals(author, other.author) && Objects.equals(bookId,
    other.bookId)
145                 && Objects.equals(bookImage, other.bookImage) && Objects.equals(category,
    other.category)
146                 && Objects.equals(condition, other.condition) &&
    Objects.equals(description, other.description)
147                 && Objects.equals(fileName, other.fileName) && Objects.equals(name,
    other.name)
148                 && Objects.equals(publisher, other.publisher) &&
    Objects.equals(releaseDate, other.releaseDate)
149                 && unitPrice == other.unitPrice && unitsInStock == other.unitsInStock;
150     }
151 }
152
```

src\main\java\com\springmvc\controller\BookController.java

```java
package com.springmvc.controller;

import java.io.File;
import java.io.IOException;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.MatrixVariable;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;

import com.springmvc.domain.Book;
import com.springmvc.exception.BookIdException;
import com.springmvc.exception.CategoryException;
import com.springmvc.service.BookService;

import jakarta.servlet.http.HttpServletRequest;

@Controller
@RequestMapping("/books") // requestbooklist와 합집합일 때 실행
public class BookController {
    @Value("${uploadPath}")
    private String uploadPath;

    private final Logger log = LoggerFactory.getLogger(BookController.class);
    //bookcontroller로 선언된 정책을 적용 (바꿀 일 없어final로 선언)

    @Autowired
    private BookService bookService;

    @GetMapping //get 요청만 처리 requestMapping에서 변경
    public String requestBookList(Model model) {
        this.log.debug("reauestBookList");

        List<Book> list = this.bookService.getAllBookList();
```

```
52          model.addAttribute("bookList", list);
53
54          return "books";// servlet-context에 있음
55      }
56
57      @GetMapping("/all")
58      public ModelAndView requestAllBooks() {
59          List<Book> list = this.bookService.getAllBookList();
60
61          ModelAndView mav = new ModelAndView();
62          mav.addObject("bookList", list);
63          mav.setViewName("books");
64
65          return mav;
66      }
67
68      @GetMapping("/{category}")
69      public String requestBooksByCategory(@PathVariable("category") String bookCategory,
    Model model) {
70          List<Book> booksByCategory = this.bookService.getBookListByCategory(bookCategory);
71          if (booksByCategory == null || booksByCategory.isEmpty()) {
72              throw new CategoryException();
73          }
74          model.addAttribute("bookList", booksByCategory);
75          return "books";
76      }
77
78      @GetMapping("/filter/{bookFilter}")
79      public String requestBooksByFilter(@MatrixVariable(pathVar = "bookFilter") Map<String,
    List<String>> bookFilter,
80              Model model) {
81          Set<Book> booksByFilter = this.bookService.getBookListByFilter(bookFilter);
82          model.addAttribute("bookList", booksByFilter);
83
84          return "books";
85      }
86
87      @GetMapping("/book")
88      public String requestBooksById(@RequestParam("id") String bookId, Model model) {
89          Book bookById = this.bookService.getBookById(bookId);
90          model.addAttribute("book", bookById);
91
92          return "book";
93      }
94
95      //get요청으로 add가 왔을때 addBook.jsp를 호출하는 기능
96      @GetMapping("/add")
97      public String requestAddBookForm(@ModelAttribute("NewBook") Book book) {
98          return "addBook"; //.jsp(뷰)로 이동: 기본 방식 포워드
99      }
100
101     @PostMapping("/add")
102     public String submitAddNewBook(@ModelAttribute("NewBook") Book book) {
103         MultipartFile bookImage = book.getBookImage();
```

```java

        if (bookImage != null && bookImage.isEmpty() == false) {
            String saveName = bookImage.getOriginalFilename();
            File saveFile = new File(this.uploadPath, saveName);
            try {
                bookImage.transferTo(saveFile);
                book.setFileName(saveName);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        this.bookService.setNewBook(book);
        return "redirect:/books"; // 리다이렉트
    }

    @ModelAttribute
    public void addAttribute(Model model) {
        model.addAttribute("addTitle", "신규 도서 등록");
    }

    @InitBinder // 어노테이션 추가
    public void initBinder(WebDataBinder binder) {
        binder.setAllowedFields("bookId", "name", "unitPrice", "author", "description",
"publisher", "category",
                "unitsInStock", "releaseDate", "condition", "bookImage");
    }

    @ExceptionHandler(value = { BookIdException.class })
    public ModelAndView handlerException(HttpServletRequest req, BookIdException
exception) {
        ModelAndView mav = new ModelAndView();
        mav.addObject("invalidBookId", exception.getBookId());
        mav.addObject("exception", exception);
        mav.addObject("url", req.getRequestURL() + "?" + req.getQueryString());
        mav.setViewName("errorBook");
        return mav;
    }
}
```

src\main\java\com\springmvc\repository\BookRepository.java

```java
package com.springmvc.repository;

import java.util.List;
import java.util.Map;
import java.util.Set;

import com.springmvc.domain.Book;

public interface BookRepository {
    List<Book> getAllBookList();
    List<Book> getBookListByCategory(String category);
    Set<Book> getBookListByFilter(Map<String, List<String>> filter);
    Book getBookById(String bookId);
    void setNewBook(Book book);
}

```

src\main\java\com\springmvc\repository\BookRepositoryImpl.java

```java
package com.springmvc.repository;

import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import com.springmvc.domain.Book;
import com.springmvc.exception.BookIdException;

@Repository
public class BookRepositoryImpl implements BookRepository {
    private JdbcTemplate template;

    public BookRepositoryImpl() { // 기본 생성자
    }

    @Autowired
    public void setJdbcTempate(JdbcTemplate template) {
        this.template =template;
    }

    @Override
    public List<Book> getAllBookList() {
        String sql ="select b_bookId, b_name, b_unitPrice, b_author,b_description,"
                +"b_publisher, b_category, b_unitsInStock, b_releaseDate, b_condition, b_fileName from book";

        List<Book> listOfBooks= this.template.query(sql,new BookRowMapper());

        return listOfBooks;
    }
    @Override
    public List<Book> getBookListByCategory(String category) {
        String sql ="select b_bookId, b_name, b_unitPrice, b_author,b_description,"
                +"b_publisher, b_category, b_unitsInStock, b_releaseDate, b_condition, b_fileName from book"
                +" where b_category like concat('%',?,'%')";

        List<Book> listOfBook = this.template.query(sql, new BookRowMapper(), category);

        return listOfBook;
    }

    @Override
    public Set<Book> getBookListByFilter(Map<String, List<String>> filter) {
        Set<Book> booksByPublish = new HashSet<>();
        Set<Book> booksByCategory = new HashSet<>();

        Set<String> booksByFilter =filter.keySet();

        if(booksByFilter.contains("publisher")){
            for(int j=0; j<filter.get("publisher").size();j++) {
                String publisherName = filter.get("publisher").get(j);
                String sql ="select b_bookId, b_name, b_unitPrice, b_author,b_description,"
                        +"b_publisher, b_category, b_unitsInStock, b_releaseDate, b_condition, b_fileName from book"
                        +" where b_publisher like concat('%',?,'%')";

                booksByPublish.addAll(this.template.query(sql, new BookRowMapper(),publisherName));
            }
        }
        if(booksByFilter.contains("category")){
            for(int j=0; j<filter.get("category").size();j++) {
                String categoryName = filter.get("category").get(j);
                String sql ="select b_bookId, b_name, b_unitPrice, b_author,b_description,"
                        +"b_publisher, b_category, b_unitsInStock, b_releaseDate, b_condition, b_fileName from book"
                        +" where b_category like concat('%',?,'%')";

                booksByCategory.addAll(this.template.query(sql, new BookRowMapper(),categoryName));
                } // 중복 결과는 제외하고 booksByCategory에 없는 내용만 들어간다.
            }
        booksByCategory.retainAll(booksByPublish);

```

```java
        return booksByCategory;
    }

    @Override
    public Book getBookById(String bookId) {
        Book bookInfo = null;

        // 1. 해당 bookId가 존재하는지 먼저 count로 확인
        String sql = "select count(*) from book where b_bookId = ?";
        int rowCount = this.template.queryForObject(sql, Integer.class, bookId);
        // → queryForObject: 결과 값이 1개 일 때 사용 (여러개: queryForList)
        // → 반환: 해당 bookId를 가진 책이 몇 개인지 (0 또는 1)
        // 2nd 파라미터: 결과 타입클래스(int), 3rd 파라미터: 쿼리문의 bookid

        // 2. 존재하면, 실제 책 정보(bookinfo)에 바인딩
        if(rowCount != 0) {
            sql ="select b_bookId, b_name, b_unitPrice, b_author,b_description,"
                    +"b_publisher, b_category, b_unitsInStock, b_releaseDate, b_condition, b_fileName from book"
                    +" where b_bookId = ?";
            bookInfo = this.template.queryForObject(sql, new BookRowMapper(), bookId);
            // 2nd 파라미터: BookRowMapper 객체→ 북 인스턴스 만들어 반환(Book), 3rd 파라미터: 쿼리문의 bookid
        }

        // 3. 그래도 null이면 예외 발생
        if(bookInfo == null) {
            throw new BookIdException(bookId);
            }
        return bookInfo;
    }

    public void setNewBook(Book book) {
        String sql ="insert into book(b_bookId, b_name, b_unitPrice, b_author,b_description,"
                +"b_publisher, b_category, b_unitsInStock, b_releaseDate, b_condition, b_fileName)"
                +" values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

        this.template.update(sql, book.getBookId(), book.getName()
                , book.getUnitPrice(), book.getAuthor(), book.getDescription()
                , book.getPublisher(), book.getCategory(), book.getUnitsInStock()
                , book.getReleaseDate(), book.getCondition(), book.getFileName());
    }
}
```

src\main\java\com\springmvc\repository\BookRowMapper.java

```java
package com.springmvc.repository;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;
import com.springmvc.domain.Book;

public class BookRowMapper implements RowMapper<Book>{// book파라미터

    @Override
    public Book mapRow(ResultSet rs, int rowNum) throws SQLException {
        // next()메소드 호출 할 필요가 없다
        Book book =new Book(); // Book 클래스의 인스턴스 생성, 참조 변수 book에 저장
        book.setBookId(rs.getString("b_bookId"));
        book.setName(rs.getString("b_name"));
        book.setUnitPrice(rs.getInt("b_unitPrice"));
        book.setAuthor(rs.getString("b_author"));
        book.setDescription(rs.getString("b_description"));
        book.setPublisher(rs.getString("b_publisher"));
        book.setCategory(rs.getString("b_category"));
        book.setUnitsInStock(rs.getLong("b_unitsInStock"));
        book.setReleaseDate(rs.getString("b_releaseDate"));
        book.setCondition(rs.getString("b_condition"));

        book.setFileName(rs.getString("b_fileName"));
        return book;
    }
}
```

src\main\java\com\springmvc\service\BookService.java

```java
package com.springmvc.service;

import java.util.List;
import java.util.Map;
import java.util.Set;

import com.springmvc.domain.Book;

public interface BookService {
    List<Book> getAllBookList();
    List<Book> getBookListByCategory(String category);
    Set<Book> getBookListByFilter(Map<String, List<String>> filter);
    Book getBookById(String bookId);
    void setNewBook(Book book);
}

```

Selected files

3 printable files

src₩main₩resources₩db₩create-db.sql
src₩main₩resources₩db₩insert-db.sql
src₩main₩resources₩application.properties

src\main\resources\db\create-db.sql

```
DROP TABLE book;

CREATE TABLE IF NOT EXISTS book(
 b_bookId VARCHAR(10) NOT NULL,
 b_name VARCHAR(30),
 b_unitPrice INTEGER,
 b_author VARCHAR(50),
 b_description TEXT,
 b_publisher VARCHAR(20),
 b_category VARCHAR(20),
 b_unitsInStock LONG,
 b_releaseDate VARCHAR(20),
 b_condition VARCHAR(20),
 b_fileName VARCHAR(20),
 PRIMARY KEY (b_bookId)
 );
```

src\main\resources\db\insert-db.sql

```
DELETE FROM book;
INSERT INTO book VALUES('ISBN1234', 'C# 교과서', 30000,'박용준', 'C# 교과서는 생애 첫 프로그래밍 언어로 C#을 시작하는 독자를 대상으로 한다. 특히 응용 프로그래머를 위한 C# 입문서로, C#을 사용하여 게임(유니티), 웹, 모바일, IoT 등을 개발할때 필요한 C# 기초 문법을 익히고 기본기를 탄탄하게 다지는 것이 목적이다.','길벗','IT전문서', 1000,'2020/05/29','', 'ISBN1234.png');
INSERT INTO book VALUES('ISBN1235', 'Node.js 교과서', 36000, '조현영', '이책은 프런트부터 서버, 데이터베이스, 배포까지 아우르는 광범위한 내용을 다룬다. 군더더기 없는 직관적인 설명으로 기본 개념을 확실히 이해하고, 노드의 기능과 생태계를사용해보면서 실제로 동작하는 서버를 만들어보자. 예제와 코드는 최신 문법을 사용했고 실무에 참고하거나 당장 적용할 수 있다.','길벗','IT전문서',1000,'2020/07/25','', 'ISBN1235.png');
INSERT INTO book VALUES('ISBN1236', '어도비 XD CC 2020', 25000, '김두한', '어도비 XD 프로그램을 통해 UI/UX 디자인을 배우고자 하는 예비 디자이너의 눈높이에맞게 기본적인 도구를 활용한 아이콘 디자인과 웹&앱 페이지 디자인, UI 디자인, 앱 디자인에 애니메이션과 인터랙션을 적용한 프로토타이핑을 학습합니다.', '길벗', 'IT활용서', 1000, '2019/05/29', '', 'ISBN1236.png');
```

src\main\resources\application.properties

```
uploadPath=C:/webjavaapp/2023136097/upload
```

src\main\webapp\WEB-INF\views\errorBook.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="jakarta.tags.core"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link href="<c:url value="/resources/css/bootstrap.css"/>" rel="stylesheet">
<title>예외 처리</title>
</head>
<body>
    <nav class="navbar navbar-expand navbar-dark bg-dark">
        <div class="container">
            <div class="navbar-header">
                <a class="navbar-brand" href="../home">Home</a>
            </div>
        </div>
    </nav>
    <div class="jumbotron">
        <div class="container">
            <h2 class="alert alert-danger">
                해당 도서가 존재하지 않습니다.<br /> 도서ID : ${invalidBookId}
            </h2>
        </div>
    </div>
    <div class="container">
        <p>${url}</p>
        <p>${exception}</p>
    </div>
    <div class="container">
        <p>
            <a href="<c:url value="/books" />" class="btn btn-secondary">도서목록 &raquo;
</a>
        </p>
    </div>
</body>
</html>
```

src\main\webapp\WEB-INF\views\errorCommon.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="jakarta.tags.core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link href="<c:url value="/resources/css/bootstrap.css"/>"
    rel="stylesheet">
<title>예외 처리</title>
</head>
<body>
    <nav class="navbar navbar-expand navbar-dark bg-dark">
        <div class="container">
            <div class="navbar-header">
                <a class="navbar-brand" href="../home">Home</a>
            </div>
        </div>
    </nav>
    <div class="jumbotron">
        <div class="container">
            <h2 class="alert alert-danger">요청한 도서가 존재하지 않습니다.</h2>
        </div>
    </div>
    <div class="container">
        <p>${exception}</p>
    </div>
    <div class="container">
        <p>
            <a href="<c:url value="/books" />" class="btn btn-secondary">도서목록
                &raquo;</a>
        </p>
    </div>
</body>
</html>

```

Selected files

3 printable files

src₩main₩java₩com₩springmvc₩exception₩BookIdException.java
src₩main₩java₩com₩springmvc₩exception₩CategoryException.java
src₩main₩java₩com₩springmvc₩exception₩CommonExceptionAdvice.java

src\main\java\com\springmvc\exception\BookIdException.java

```java
package com.springmvc.exception;

public class BookIdException extends RuntimeException {
    private String bookId;

    public BookIdException(String bookId) {
        this.bookId = bookId;
    }

    public String getBookId() {
        return this.bookId;
    }
}
```

src\main\java\com\springmvc\exception\CategoryException.java

```java
package com.springmvc.exception;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.NOT_FOUND, reason = "요청한 도서 분야를 찾을 수 없습니다.")
public class CategoryException extends RuntimeException {
    public CategoryException() {
        super();
    }
    // 그 밖에 생성자
}
```

src\main\java\com\springmvc\exception\CommonExceptionAdvice.java

```java
package com.springmvc.exception;

import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.servlet.ModelAndView;

@ControllerAdvice
public class CommonExceptionAdvice {
    @ExceptionHandler(RuntimeException.class)
    private ModelAndView handleExceptionCommon(Exception e) {
        e.printStackTrace();
        ModelAndView mav = new ModelAndView();
        mav.addObject("exception", e);
        mav.setViewName("errorCommon");
        return mav;
    }
}
```

src\main\resources\log4j2.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration xmlns="https://logging.apache.org/xml/ns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
    xsi:schemaLocation="https://logging.apache.org/xml/ns
                                https://logging.apache.org/xml/ns/log4j-config-2.xsd">
    <Appenders>
        <Console name="console">
            <PatternLayout
                pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p %c{-10}:%L - %m%n" />
        </Console>

        <RollingFile name="monitor"
            fileName="C:/webjavaapp/2023136097/logs/monitor.log"
            filePattern="C:/webjavaapp/2023136097/logs/monitor.%d{yyyy-MM-dd-hh-mm}.log">
            <LevelRangeFilter minLevel="DEBUG" maxLevel="ERROR" />
            <PatternLayout
                pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p %{-10}:%L -%m%n" />
            <Policies>
                <SizeBasedTriggeringPolicy size="10KB" />
                <TimeBasedTriggeringPolicy interval="1" />
            </Policies>
        </RollingFile>

    </Appenders>
    <Loggers>
        <Root level="INFO" additivity="false">
            <AppenderRef ref="console" />
        </Root>
        <Logger name="com.springmvc" level="DEBUG">
            <AppenderRef ref="monitor" />
        </Logger>
        <Logger name="org.springframework.core" level="INFO" />
        <Logger name="org.springframework.beans" level="INFO" />
        <Logger name="org.springframework.context" level="INFO" />
        <Logger name="org.springframework.web" level="DEBUG" />
        <Logger name="org.springframework.security" level="DEBUG" />
        <Logger name="org.springframework.jdbc" level="DEBUG" />
    </Loggers>
</Configuration>
```

src\main\webapp\WEB-INF\views\login.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<link href="<c:url value="/resources/css/bootstrap.css"/>" rel="stylesheet">
<title>로그인</title>
</head>
<body>
    <nav class="navbar navbar-expand navbar-dark bg-dark">
        <div class="container">
            <div class="navbar-header">
                <a class="navbar-brand" href="./home">Home</a>
            </div>
        </div>
    </nav>
    <div class="jumbotron">
        <div class="container">
            <h1 class="display-3">로그인</h1>
        </div>
    </div>
    <div class="container col-md-4">
        <div class="text-center">
            <h3 class="form-signin-heading">Please sign in</h3>
        </div>
        <c:if test="${not empty error}">
            <div class="alert alert-danger">
                UserName과 Password가 올바르지 않습니다.<br />
            </div>
        </c:if>
        <form class="form-signin" action="<c:url value="/login"/>" method="post">
            <div class="form-group row">
                <input type="text" name="username" class="form-control" placeholder="User
Name" required autofocus>
            </div>
            <div class="form-group row">
                <input type="password" name="password" class="form-control"
placeholder="Password" required>
            </div>
            <div class="form-group row">
                <button class="btn btn-lg btn-success btn-block" type="submit">로그인
</button>
                <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"
/>
            </div>
        </form>
    </div>
</body>
</html>
```

src\main\java\com\springmvc\controller\LoginController.java

```java
package com.springmvc.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class LoginController { //로그인 뷰 처리

    //  로그인 처리
    @GetMapping("/login")
    public String login() {
        return "login";
    }

    @GetMapping("/loginfailed")
    public String loginerror(Model model) {
        model.addAttribute("error", "true");
        return "login";
    }

    //  로그아웃 처리
    @GetMapping("/logout")
    public String logout(Model model) {
        return "login";
    }
}
```

src\main\java\com\springmvc\interceptor\MonitoringInterceptor.java

```java
package com.springmvc.interceptor;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.util.StopWatch; //요청 시작부터 끝까지의 시간
import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class MonitoringInterceptor implements HandlerInterceptor {
    private final Logger log = LoggerFactory.getLogger(MonitoringInterceptor.class);
    ThreadLocal<StopWatch> stopWatchLocal = new ThreadLocal<StopWatch>();

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
            throws Exception {
        StopWatch stopWatch = new StopWatch(handler.toString());
        stopWatch.start(handler.toString());

        this.stopWatchLocal.set(stopWatch);
        // {}:파라미터로 치환가능 ,다음에 나오는 값이 순서대로 매칭된다.
        this.log.info("접근한 URL 경로: {}", this.getURLPath(request));
        this.log.info("요청처리시작시간:{}", this.getCurrentTime());

        return true;
    }

    private String getURLPath(HttpServletRequest request) {
        String currentPath = request.getRequestURI();
        String queryString = request.getQueryString();
        queryString = queryString == null ? "" : "?" + queryString;
        return currentPath + queryString;
    }

    private String getCurrentTime() {
        DateFormat formatter = new SimpleDateFormat("yyy/MM/dd HH:mm:ss");
        Calendar calendar = Calendar.getInstance();
        calendar.setTimeInMillis(System.currentTimeMillis());
        return formatter.format(calendar.getTime());
    }

    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,
            ModelAndView modelAndView) throws Exception {

        this.log.info("요청처리 종료 시간: {}", this.getCurrentTime());
    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler,
Exception ex)
            throws Exception {
        StopWatch stopWatch = this.stopWatchLocal.get();
        stopWatch.stop();
        this.log.info("요청 처리 소요시간: {} ms", stopWatch.getTotalTimeMillis());
        this.stopWatchLocal.set(null);
    }
}
```

pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>webjava</groupId>
    <artifactId>book-store</artifactId>
    <version>0.0.1</version>
    <packaging>war</packaging>

    <name>book-store</name>
    <description>Book Store Project</description>

    <properties>
        <java-version>21</java-version>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <!-- Spring MVC -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>6.2.1</version>
        </dependency>

        <!-- Servlet API -->
        <dependency>
            <groupId>jakarta.servlet</groupId>
            <artifactId>jakarta.servlet-api</artifactId>
            <version>6.0.0</version>
            <scope>provided</scope>
        </dependency>

        <!-- JSP API -->
        <dependency>
            <groupId>jakarta.servlet.jsp</groupId>
            <artifactId>jakarta.servlet.jsp-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>

        <!-- Log4j + SLF4J -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j2-impl</artifactId>
            <version>2.24.3</version>
        </dependency>
```

```
52
53          <!-- JSTL API -->
54          <dependency>
55              <groupId>jakarta.servlet.jsp.jstl</groupId>
56              <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
57              <version>3.0.0</version>
58          </dependency>
59
60          <!-- JSTL Implementation -->
61          <dependency>
62              <groupId>org.glassfish.web</groupId>
63              <artifactId>jakarta.servlet.jsp.jstl</artifactId>
64              <version>3.0.1</version>
65          </dependency>
66
67          <!-- H2 Database -->
68          <dependency>
69              <groupId>com.h2database</groupId>
70              <artifactId>h2</artifactId>
71              <version>2.3.232</version>
72          </dependency>
73
74          <!-- Spring JDBC -->
75          <dependency>
76              <groupId>org.springframework</groupId>
77              <artifactId>spring-jdbc</artifactId>
78              <version>6.2.1</version>
79          </dependency>
80
81          <!-- Spring Security -->
82          <dependency>
83              <groupId>org.springframework.security</groupId>
84              <artifactId>spring-security-web</artifactId>
85              <version>6.4.2</version>
86          </dependency>
87
88          <dependency>
89              <groupId>org.springframework.security</groupId>
90              <artifactId>spring-security-config</artifactId>
91              <version>6.4.2</version>
92          </dependency>
93
94          <dependency>
95              <groupId>org.springframework.security</groupId>
96              <artifactId>spring-security-taglibs</artifactId>
97              <version>6.4.2</version>
98          </dependency>
99      </dependencies>
100 </project>
101
```

src\main\webapp\WEB-INF\spring\security-context.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans
    xmlns="http://www.springframework.org/schema/security"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/security
        http://www.springframework.org/schema/security/spring-security.xsd">

    <!-- HTTP 보안 설정 -->
    <http use-expressions="true">
        <intercept-url pattern="/books/add" access="hasAuthority('ROLE_ADMIN')" />
        <intercept-url pattern="/**" access="permitAll" />

        <!-- 로그인 설정 -->
        <form-login
            login-page="/login"
            default-target-url="/books/add"
            authentication-failure-url="/loginfailed"
            username-parameter="username"
            password-parameter="password" />
        <csrf />

        <!-- 로그아웃 설정 -->
        <logout logout-success-url="/logout" />
    </http>

    <!-- 인증 관리자 설정 -->
    <authentication-manager>
        <authentication-provider user-service-ref="customUserService">
            <password-encoder ref="customPasswordEncoder" />
        </authentication-provider>
    </authentication-manager>

    <!-- 이전에 쓰던 부분이다. 아래코드에서 위코드로 코드확장했음
        <authentication-manager>
        <authentication-provider user-service-ref="customUserService">
            <user-service>
                <user name="Admin" password="{noop}Admin1234"
                    authorities="ROLE_ADMIN" />
            </user-service>
        </authentication-provider>
    </authentication-manager>
     -->

    <!-- 커스텀 빈 등록 -->
    <beans:bean id="customUserService" class="com.springmvc.service.CustomUserDetailsService" />
    <beans:bean id="customPasswordEncoder" class="com.springmvc.service.CustomPasswordEncoder" />
    <beans:bean id="mvcHandlerMappingIntrospector"
class="org.springframework.web.servlet.handler.HandlerMappingIntrospector" />

</beans:beans>
```

src\main\webapp\WEB-INF\spring\servlet-context.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans
    xmlns="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
                        http://www.springframework.org/schema/mvc/spring-mvc.xsd
                        http://www.springframework.org/schema/beans
                        http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://www.springframework.org/schema/context
                        http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- 다국어 메시지 설정 -->
    <beans:bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessag-
eSource">
        <beans:property name="basename" value="messages" />
        <beans:property name="defaultEncoding" value="UTF-8" />
    </beans:bean>

    <!-- 인터셉터 설정 -->
    <interceptors>
        <beans:bean class="com.springmvc.interceptor.MonitoringInterceptor" />
        <beans:bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
            <beans:property name="paramName" value="language" />
        </beans:bean>
    </interceptors>

    <!-- 로케일 설정 -->
    <beans:bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolver">
        <beans:property name="defaultLocale" value="UTF-8" />
    </beans:bean>

    <!-- 프로퍼티 파일 로딩 -->
    <context:property-placeholder location="classpath:application.properties" />

    <!-- 어노테이션 기반 MVC 활성화 -->
    <annotation-driven enable-matrix-variables="true" />

    <!-- 컴포넌트 스캔 -->
    <context:component-scan base-package="com.springmvc.*" />

    <!-- 정적 리소스 매핑 -->
    <resources mapping="/resources/**" location="/resources/" />

    <!-- ViewResolver 설정 -->
    <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />
        <beans:property name="suffix" value=".jsp" />
    </beans:bean>

    <!-- JDBC Template -->
    <beans:bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <beans:property name="dataSource" ref="dataSource" />
    </beans:bean>

    <!-- 파일 업로드 처리 -->
    <beans:bean id="multipartResolver"
class="org.springframework.web.multipart.support.StandardServletMultipartResolver" />
</beans:beans>
```

src\main\webapp\WEB-INF\web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                             https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
         metadata-complete="false"  version="6.0">

    <!-- 루트 스프링 설정 파일 -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/spring/root-context.xml
            /WEB-INF/spring/security-context.xml
        </param-value>
    </context-param>
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!-- DispatcherServlet 설정 -->
    <servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>

        <!-- 파일 업로드 관련 설정 -->
        <multipart-config>
            <location>C:\\webjavaapp\\2021136073\\upload</location>
            <max-file-size>20971520</max-file-size>
            <file-size-threshold>20971520</file-size-threshold>
            <max-request-size>41942040</max-request-size>
        </multipart-config>
    </servlet>

    <servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <!-- H2 콘솔 서블릿 설정 -->
    <servlet>
        <servlet-name>H2Console</servlet-name>
        <servlet-class>org.h2.server.web.JakartaWebServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>H2Console</servlet-name>
```

```
        <url-pattern>/console/*</url-pattern>
    </servlet-mapping>

    <!-- Spring Security 필터 -->
    <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 인코딩 필터 -->
    <filter>
        <filter-name>encodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

Selected files

2 printable files

src₩main₩resources₩messages_en.properties
src₩main₩resources₩messages_ko.properties

src\main\resources\messages_en.properties

```
# IOS로 해도 상관없지만 한국어 파일과 인코딩타입 일치를 위해 propertise에서 utf-8로 변경해주기
addBook.form.title.label=Book Addition
addBook.form.subtitle.label=New Book Registration
addBook.form.bookId.label=Book ID
addBook.form.name.label=Name
addBook.form.unitPrice.label=Unit Price
addBook.form.author.label=Author
addBook.form.description.label=Description
addBook.form.publisher.label=Publisher
addBook.form.category.label=Category
addBook.form.unitsInStock.label=Units in Stock
addBook.form.releaseDate.label=Release Date
addBook.form.condition.label=Condition
addBook.form.bookImage.label=Book Image
addBook.form.button.label=Addition
```

src\main\resources\messages_ko.properties

```
# 한국어 파일이기 때문에 propertise에서 utf-8로 변경해주기
addBook.form.title.label=도서등록
addBook.form.subtitle.label=신규도서등록
addBook.form.bookId.label=도서ID
addBook.form.name.label=도서명
addBook.form.unitPrice.label=가격
addBook.form.author.label=저자
addBook.form.description.label=상세정보
addBook.form.publisher.label=출판사
addBook.form.category.label=분야
addBook.form.unitsInStock.label=재고수
addBook.form.releaseDate.label=출판일
addBook.form.condition.label=상태
addBook.form.bookImage.label=도서이미지
addBook.form.button.label=등록
```

src\main\java\com\springmvc\service\BookServiceImpl.java

```java
package com.springmvc.service;

import java.util.List;
import java.util.Map;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.springmvc.domain.Book;
import com.springmvc.repository.BookRepository;

@Service // 스프링이 자동으로 Bean으로 등록해서 DI(의존성 주입) 대상이 되도록 관리
public class BookServiceImpl implements BookService {

    @Autowired // 스프링이 관리하는 Bean 객체를 자동으로 주입(의존성 주입)해주는 어노테이션
    //레파지토리보다 impl이 먼저 생성되어야함 객체화 되는 시점에 이미 bean 레파지토리에 등록 실행
    private BookRepository bookRepository;

    @Override // 이 시점에서 이미 할당 완료
    public List<Book> getAllBookList() {
        return this.bookRepository.getAllBookList();
    }

    @Override
    public List<Book> getBookListByCategory(String category) {
        List<Book> listOfBook = this.bookRepository.getBookListByCategory(category);

        return listOfBook;
    }

    @Override
    public Set<Book> getBookListByFilter(Map<String, List<String>> filter) {
        Set<Book> booksByFilter = this.bookRepository.getBookListByFilter(filter);

        return booksByFilter;
    }

    @Override
    public Book getBookById(String bookId) {
        Book bookInfo = this.bookRepository.getBookById(bookId);

        return bookInfo;
    }

    @Override
    public void setNewBook(Book book) {
        this.bookRepository.setNewBook(book);
    }
}

```