ＳＴＥＰ０５

Ｓｐｒｉｎｇフレームワーク入門

コードで理解するＳｐｒｉｎｇの世界

Spring JDBCによる書籍テーブル更新

目次

[1 書籍登録フォーム画面をきちんと実装する 1](#_Toc428282202)

[1.1 書籍登録フォーム画面の再実装 1](#_Toc428282204)

[1.2 画面表示テスト 2](#_Toc428282205)

[2 書籍情報登録処理をきちんと実装する 3](#_Toc428282206)

[2.1 書籍登録Ｄａｏクラスをきちんと作成 3](#_Toc428282208)

[2.2 書籍登録サービスクラスをきちんと作成 5](#_Toc428282209)

[2.3 書籍登録POSTリクエストを受け付けるコントローラメソッドを追加 7](#_Toc428282210)

[2.4 画面表示テスト 9](#_Toc428282211)

[3 書籍登録結果画面への遷移 10](#_Toc428282212)

[3.1 書籍登録結果画面に遷移するように変更 10](#_Toc428282214)

[3.2 画面表示テスト 11](#_Toc428282215)

# 書籍登録フォーム画面をきちんと実装する



## 書籍登録フォーム画面の再実装

１）書籍情報登録フォーム画面コントローラメソッドの修正

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.web.controller.BookController.java |
| ● 56行目付近（BookController#addBookFrom メソッドを修正） |
| 56  /\*\*  \* 書籍登録フォーム画面コントローラ  \*  \* @param model Spring MVC Modelオブジェクト  \* @return 画面JSP名  \*/  @RequestMapping(value = "/addbookform", method = RequestMethod.GET)  public String addBookForm(Model model) throws Exception {    logger.debug("addbookform() start");    // 空の書籍情報をモデルに登録  Book book = new Book();  model.addAttribute("book", book);    // 画面表示に addbookform.jsp を呼び出す  return "addbookform";  } |

２）書籍登録画面View（addbookform.jsp）をきちんと修正

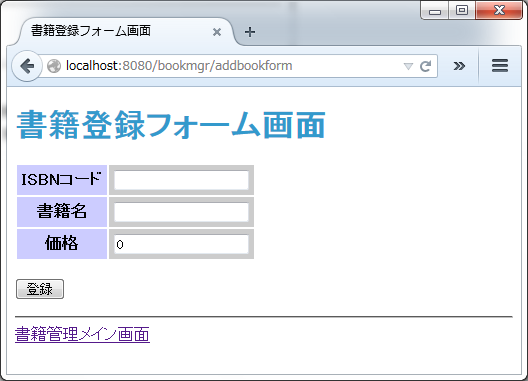
|  |
| --- |
| /src/main/webapp/WEB-INF/views/addbookform.jsp |
| <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <link rel="stylesheet" type="text/css" href="css/bookmgr.css" />  <title>画面登録フォーム画面</title>  </head>  <body>  <h1>書籍登録フォーム画面</h1>  <p>  <form:form action="addbook" method="POST" modelAttribute="book">  <table>  <tr>  <th>ISBNコード</th>  <td><form:input path="isbn"/></td>  </tr><tr>  <th>書籍名</th>  <td><form:input path="name"/></td>  </tr><tr>  <th>価格</th>  <td><form:input path="price"/></td>  </tr>  </table>  <br>  <input type="submit" value="登録" />  </form:form>  </p>  <hr />  <a href="main">書籍管理メイン画面</a>  </body>  </html> |

Spring Formタグの説明は、以下のサイトが詳しい

http://kuwalab.hatenablog.jp/entry/20130118/p1

## 画面表示テスト

ここで、書籍登録画面（フォーム）が表示されるかどうかをチェックすること。



なお、「登録」ボタンを押すと書籍情報の登録リクエストが発生するが、まだ当該処理を実装していないので 404エラーとなる。（これより実装します）

# 書籍情報登録処理をきちんと実装する

書籍情報登録処理を追加。登録を実行し登録結果を表示する画面に遷移させる



## 書籍登録Ｄａｏクラスをきちんと作成

１）書籍登録Ｄａｏクラスインターフェース（AddBookDao.java）新規作成＋コーディング

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.biz.dao.AddBookDao.java （新規作成） |
| package jp.sample.bookmgr.biz.dao;  import org.springframework.dao.DataAccessException;  import jp.sample.bookmgr.biz.domain.Book;  /\*\*  \* 書籍登録データアクセスクラスインターフェース  \*  \* @author 長住@NTT-AT  \* @version 1.0  \*/  public interface AddBookDao {  /\*\*  \* 指定された書籍データをデータベースに登録して永続化するメソッド  \*  \* @param Book 書籍情報  \* @return int 登録された書籍データ数  \*/  public int addBook(Book book) throws DataAccessException;  } |

２）書籍登録Ｄａｏ実装クラス（AddBookDaoSpringJdbc.java）を新規作成＋コーディング

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.biz.dao.AddBookDaoSpringJdbc.java （新規作成） |
| package jp.sample.bookmgr.biz.dao;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.dao.DataAccessException;  import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;  import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;  import org.springframework.stereotype.Repository;  import jp.sample.bookmgr.biz.domain.Book;  /\*\*  \* 書籍登録データアクセス実装クラス  \*  \* @author 長住@NTT-AT  \* @version 1.0  \*/  @Repository // DAOとしてDI可能というアノテーションを宣言  public class AddBookDaoSpringJdbc implements AddBookDao {  /\*\*  \* JDBC制御クラス  \*/  @Autowired // インジェクション  private NamedParameterJdbcTemplate template;    /\*\*  \* 指定された書籍データをデータベースに登録して永続化するメソッド  \*  \* @param book 書籍情報  \* @return int 登録された書籍データ数  \*/  @Override  public int addBook(Book book) throws DataAccessException {    // 書籍登録SQLのプレースホルダーを設定  MapSqlParameterSource pmap = new MapSqlParameterSource();  pmap.addValue("ISBN", book.getIsbn());  pmap.addValue("NAME", book.getName());  pmap.addValue("PRICE", book.getPrice());  // 書籍登録SQL文  String sql = "INSERT INTO book (isbn, name, price) VALUES (:ISBN, :NAME, :PRICE)";  // データベースに書籍を登録する  int count = template.update(sql, pmap);  return count;  }  } |

## 書籍登録サービスクラスをきちんと作成

１）書籍登録サービスクラスインターフェース（AddBookService.java）新規作成＋コーディング

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.biz.service.AddBookService.java （新規作成） |
| package jp.sample.bookmgr.biz.service;  import jp.sample.bookmgr.biz.domain.Book;  /\*\*  \* 書籍登録サービスインターフェース  \*  \* @author 長住@NTT-AT  \* @version 1.0  \*/  public interface AddBookService {    /\*\*  \* 書籍登録サービスを実行する  \*  \* @param Book 書籍情報  \*/  public void addBook(Book book) throws Exception ;  } |

２）書籍登録サービス実装クラス（AddBookServiceImpl.java）新規作成＋コーディング

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.biz.service.AddBookServiceImpl.java （新規作成） |
| package jp.sample.bookmgr.biz.service;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import jp.sample.bookmgr.biz.dao.AddBookDao;  import jp.sample.bookmgr.biz.domain.Book;  /\*\*  \* 書籍登録サービス実装クラス  \*  \* @author 長住@NTT-AT  \* @version 1.0  \*/  @Service // サービスクラスとしてDI可能というアノテーションを宣言  public class AddBookServiceImpl implements AddBookService {  /\*\*  \* データベースに書籍登録を行うDAOクラス  \*/  @Autowired // インジェクション  private AddBookDao addBookDao;    /\*\*  \* 書籍登録サービスを実行する  \*  \* @param Book 書籍情報  \*/  @Override  public void addBook(Book book) throws Exception {  // 書籍登録を行うDAOクラスを使用して書籍情報を永続化する  addBookDao.addBook(book);  }  } |

## 書籍登録POSTリクエストを受け付けるコントローラメソッドを追加

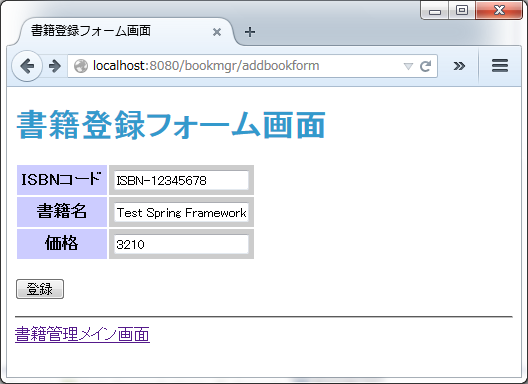
１）書籍情報画面コントロールクラスに書籍登録POSTリクエストを受け付けるメソッドを追加

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.web.controller.BookController.java |
| package jp.sample.bookmgr.web.controller;  import java.util.List;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.ModelAttribute;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import jp.sample.bookmgr.biz.domain.Book;  import jp.sample.bookmgr.biz.service.AddBookService;  import jp.sample.bookmgr.biz.service.ListBookService;  /\*\*  \* 書籍管理処理コントローラクラス  \*  \* @author 長住@NTT-AT  \* @version 1.0  \*/  @Controller // ”コントローラとしてDI可能" というアノテーションを宣言  public class BookController {  /\*\*  \* ロガー  \*/  private static final Logger logger = LoggerFactory.getLogger(BookController.class);    /\*\*  \* 書籍一覧処理を実装したビジネスロジックサービス  \*/  @Autowired // ListBookServiceオブジェクトをインジェクション  ListBookService listBookService;    /\*\*  \* 書籍登録処理を実装したビジネスロジックサービス  \*/  @Autowired // インジェクション  AddBookService addBookService;    /\*\*  \* 書籍一覧画面コントローラ  \*  \* @param model Spring MVC Modelオブジェクト  \* @return 画面JSP名  \*/  @RequestMapping(value = "/listbook", method = RequestMethod.GET)  public String listBook(Model model) throws Exception {    logger.debug("listBook() start");    // 書籍一覧取得ロジック処理  List<Book> books = listBookService.getBookList();  // 書籍一覧情報をモデルに登録  model.addAttribute("books", books);    // 画面表示に listbook.jsp を呼び出す  return "listbook";  }  /\*\*  \* 書籍登録フォーム画面コントローラ  \*  \* @param model Spring MVC Modelオブジェクト  \* @return 画面JSP名  \*/  @RequestMapping(value = "/addbookform", method = RequestMethod.GET)  public String addBookForm(Model model) throws Exception {    logger.debug("addbookform() start");    // 空の書籍情報をモデルに登録  Book book = new Book();  model.addAttribute("book", book);    // 画面表示に addbookform.jsp を呼び出す  return "addbookform";  }    80  /\*\*  \* 書籍登録処理コントローラ  \*  \* @param book Spring MVC ModelオブジェクトからバインドされたBookインスタンス  \* @return 登録結果表示JSP名  \*/  @RequestMapping(value = "/addbook", method = RequestMethod.POST)  public String addBook(@ModelAttribute("book") Book book) throws Exception {    logger.debug("addbook() start");  logger.debug("book isbn:" + book.getIsbn());  logger.debug("book name:" + book.getName());  logger.debug("book price:" + book.getPrice());    // 書籍情報登録処理  addBookService.addBook(book);    // とりあえず、メイン画面に戻るようにする  return "main";  }  } |

## 画面表示テスト

ここで、書籍登録画面から、以下のデータを投入して登録させてみる。

* ここでは、英数字のみでテストしてください。（漢字、ひらがな等は文字化けするため）



正常に登録できたでしょうか？

登録できたら、書籍一覧に移動して、今登録したデータが表示されることを確認。



日本語を入力すると文字化けする！の対策は、STEP6で行います。

# 書籍登録結果画面への遷移



## 書籍登録結果画面に遷移するように変更

１）書籍管理コントローラクラスを修正・追加

書籍登録コントロールメソッドにおける遷移先の画面Viewの変更と、遷移先URLに対応するリクエストを受け付けるメソッドの追加。

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.web.controller.BookController.java |
| ● 82行目付近（BookController.AddBook メソッドの最後を修正） |
| /\*\*  \* 書籍登録処理コントローラ  \*  \* @param book Spring MVC ModelオブジェクトからバインドされたBookインスタンス  \* @return 登録結果表示JSP名  \*/  @RequestMapping(value = "/addbook", method = RequestMethod.POST)  public String addBook(@ModelAttribute("book") Book book) throws Exception {    logger.debug("addbook() start");  logger.debug("book isbn:" + book.getIsbn());  logger.debug("book name:" + book.getName());  logger.debug("book price:" + book.getPrice());    // 書籍情報登録処理  addBookService.addBook(book);    98  // 登録結果画面 result.jsp を呼び出す  return "redirect:result";  } |
| ● 103行目付近（BookController.AddBook メソッドの次に挿入） |
| /\*\*  \* 処理結果画面表示コントローラ  \*  \* @return 登録結果表示JSP名  \*/  @RequestMapping(value = "/result", method = RequestMethod.GET)  public String result() throws Exception {  logger.info("result START");  return "result";  } |

２）書籍登録結果画面View（result.jsp）を新規作成＋コーディング

|  |
| --- |
| /src/main/webapp/WEB-INF/views/result.jsp （新規作成） |
| <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <link rel="stylesheet" type="text/css" href="css/bookmgr.css" />  <title>更新結果</title>  </head>  <body>  <h1>更新に成功しました!</h1>  <hr />  <a href="main">書籍管理メイン画面</a>  <br/>  <a href="listbook">書籍一覧画面</a>  </body>  </html> |

## 画面表示テスト

書籍登録画面→登録結果画面 に遷移することを確認。



＃ここでは、英数字のみでテストしてください。（漢字、ひらがな等は文字化けするため）

＃日本語を入力すると文字化けする！の対策は、STEP6で行います。