ＳＴＥＰ０６

Ｓｐｒｉｎｇフレームワーク入門

コードで理解するＳｐｒｉｎｇの世界

Validation 実装と日本語などの対策

目次

[1 日本語文字化け対策、およびHTML ESCAPEの設定 1](#_Toc428284317)

[1.1 日本語文字化け対策等の環境設定 1](#_Toc428284319)

[1.2 画面表示テスト 2](#_Toc428284320)

[2 バリデーション（妥当性）チェックの準備 3](#_Toc428284321)

[2.1 日本語文字でプロパティファイルを編集できるプラグインをEclipseに導入 3](#_Toc428284323)

[2.2 バリデーションを実装するライブラリ登録 （Maven） 3](#_Toc428284324)

[2.3 メッセージソース設定 （Spring定義ファイル／プロパティファイル） 4](#_Toc428284325)

[3 書籍情報登録時のバリデーション（妥当性）チェック処理を実装する 5](#_Toc428284326)

[3.1 書籍（Book）クラスにバリデーションチェック用のアノテーションを追加 5](#_Toc428284328)

[3.2 書籍登録処理のコントローラメソッドにバリデーションチェック処理を追加 6](#_Toc428284329)

[3.3 画面表示テスト 8](#_Toc428284330)

# 日本語文字化け対策、およびHTML ESCAPEの設定



## 日本語文字化け対策等の環境設定

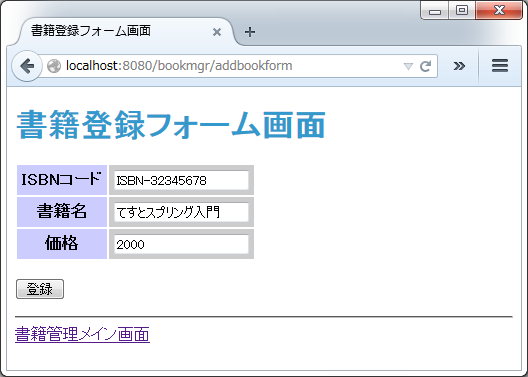
日本語文字が文字化けする対策について、Spring MVCを使用するときは、それ提供するフィルタを設定する方法が最も確実で簡単で一般的。また、通常、HTML ESCAPEの設定も併せて行う。

１）日本語文字化け対策、およびHTML ESCAPEの設定

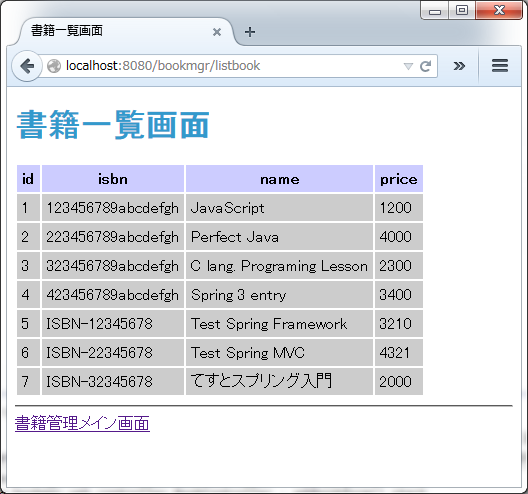
|  |
| --- |
| /src/main/webapp/WEB-INF/web.xml |
| ● 34行目付近から表示 / 追記箇所は39行目付近から |
| <servlet-mapping>  <servlet-name>appServlet</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>    39  <!-- HTML ESCAPEの設定 -->  <context-param>  <param-name>defaultHTMLEscape</param-name>  <param-value>true</param-value>  </context-param>    <!-- 日本語文字化け対策の設定 -->  <filter>  <filter-name>CharacterEncodingFilter</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>  <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>    </web-app> |

## 画面表示テスト

書籍登録画面から、以下の日本語込のデータを投入して登録させてみる。



書籍一覧で、正常に登録されることを確認。



# バリデーション（妥当性）チェックの準備



## 日本語文字でプロパティファイルを編集できるプラグインをEclipseに導入

|  |
| --- |
| ① Spring Tool Suite のインストール  Help → Eclipse MarketPlace  検索画面で[PropertiesEditor]を検索して，以下のツールをインストールする．  -----------------  Properties Editor  ----------------- |

## インストールの途中でワーニングが出るがインストールを続行してOK。

## インストール後再起動。

PropertiesEditor は必須ではないが、これがないと、日本語によるプロパティファイルの編集が面倒くさくなる！ （いちいちnative2ascii などで変換しないといけなくなる）

## バリデーションを実装するライブラリ登録 （Maven）

１）Maven設定ファイルにValidatorライブラリ（Hybernate Validator）を追記

|  |
| --- |
| /pom.xml |
| ● 58行目付近 |
| <dependency>  <groupId>org.postgresql</groupId>  <artifactId>postgresql</artifactId>  <version>9.4.1208.jre7</version>  </dependency>    64  <!-- Validation -->  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-validator</artifactId>  <version>5.2.4.Final</version>  </dependency>  <!-- Logging -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>${org.slf4j-version}</version>  </dependency> |

## Maven設定ファイルを更新するとプロジェクトがリビルドする。

## ビルドの進捗は右下に地味に表示されているだけなので見落とさないこと。

## ビルドに数秒～数分がかかる場合もあるが、終わるまで待っているほうが吉。

## メッセージソース設定 （Spring定義ファイル／プロパティファイル）

１）Spring Bean 定義ファイル（application-context-biz.xml）更新

メッセージソースとValidator Bean定義を追加する。

|  |
| --- |
| /src/main/webapp/WEB-INF/spring/application-context-biz.xml |
| ● 31行目付近 |
| <!-- JDBC Named Template Bean -->  <bean id="namedJdbcTemplate" class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">  <constructor-arg ref="dataSource" />  </bean>    36  <!-- Message Source -->  <bean id="messageSource" class="org.springframework.context.support.ReloadableResourceBundleMessageSource">  <property name="basename" value="classpath:/messages" />  </bean>  <!-- Validator -->  <bean id="validator" class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">  <property name="validationMessageSource" ref="messageSource" />  </bean>    </beans> |

２）Spring Bean 定義ファイル（application-context-web.xml）更新

<mvc:annotation-driven /> に validator="validator" を追記

|  |
| --- |
| /src/main/webapp/WEB-INF/spring/application-context-web.xml |
| ● 18行目の <mvc:annotation-driven /> に validator="validator" を追記 |
| <!-- Defines this servlet's request-processing infrastructure -->  <context:component-scan base-package="jp.sample. bookmgr.web" />    <!-- Enables the Spring MVC @Controller programming model -->  18  <mvc:annotation-driven validator="validator"/>  <!-- Resource mappings -->  <mvc:resources mapping="/resources/\*\*" location="/resources/" />  <mvc:resources mapping="/css/\*\*" location="/css/" /> |

３）メッセージを日本語で表示するためのプロパティファイルを新規作成

|  |
| --- |
| /src/main/resources/messages\_ja.properties （新規作成） |
| typeMismatch.int=数値を入力してください  javax.validation.constraints.NotNull.message={0}は必須です。  javax.validation.constraints.Size.message=サイズは{min}から{max}の間の値を入力してください。  org.hibernate.validator.constraints.NotEmpty.message={0}は必須です。  org.hibernate.validator.constraints.NotBlank.message={0}は必須です。  book.isbn=ISBNコード  book.name=書籍名  book.price=価格 |

＃エラーメッセージプロパティの記述方法は、Spring3入門の212ページ（第6章）が参考になる。

# 書籍情報登録時のバリデーション（妥当性）チェック処理を実装する



## 書籍（Book）クラスにバリデーションチェック用のアノテーションを追加

１）書籍（Book）クラスにバリデーションのアノテーションを定義

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.biz.domain.Book.java |
| package jp.sample.bookmgr.biz.domain;  import javax.validation.constraints.NotNull;  import javax.validation.constraints.Size;  import org.hibernate.validator.constraints.NotBlank;  /\*\*  \* 書籍クラス  \* @author 長住@NTT-AT  \* @version 1.0  \*/  public class Book {  /\*\*  \* 書籍ID  \*/  private int id = 0;    /\*\*  \* 書籍ISBN  \*/  @NotBlank  @Size(min=10, max=16)  private String isbn = "";    /\*\*  \* 書籍名  \*/  @NotBlank  private String name = "";    /\*\*  \* 書籍価格  \*/  @NotNull  private int price = 0;  以下、変更なし  ・・・  ・・・ |

＃バリデーションでよく使うアノテーションの種類は、Spring3入門の208ページ（第6章）が参考になる。

## 書籍登録処理のコントローラメソッドにバリデーションチェック処理を追加

１）書籍情報登録コントローラメソッドにバリデーション実行処理を追加

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.web.controller.BookController.java |
| ● 86行目付近 |
| /\*\*  \* 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  \* @param result バインディング処理時のバリデーションチェック結果  \* @return 登録結果表示JSP名  \*/  @RequestMapping(value = "/addbook", method = RequestMethod.POST)  public String addBook(@Valid @ModelAttribute("book") Book book,  BindingResult result) throws Exception {    logger.debug("addbook() start");  logger.debug("book isbn:" + book.getIsbn());  logger.debug("book name:" + book.getName());  logger.debug("book price:" + book.getPrice());    // Validation エラーが発生しているときは登録フォームに戻る  if(result.hasErrors()){  return "addbookform";  }    // 書籍情報登録処理  addBookService.addBook(book);    // 登録結果画面 result.jsp を呼び出す  return "redirect:result";  } |

＃バリデーションの実行に関する説明は、Spring3入門の213ページ（第6章）が参考になる。

２）書籍登録画面View（JSP）にエラー表示処理を追加

|  |
| --- |
| /src/main/webapp/WEB-INF/views/addbookform.jsp |
| <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>  <%@ page contentType="text/html; charset=UTF-8" %>  <%@ page pageEncoding="UTF-8" %>  <%@ page session="false" %>  <html>  <head>  <link rel="stylesheet" type="text/css" href="css/bookmgr.css" />  <title>書籍登録画面</title>  </head>  <body>  <h1>書籍登録画面</h1>  <p>  <form:form action="addbook" method="POST" modelAttribute="book">  <table>  <tr>  <th>ISBNコード</th>  <td><form:input path="isbn"/></td>  <td><form:errors path="isbn" cssStyle="color:red"/></td>  </tr><tr>  <th>書籍名</th>  <td><form:input path="name"/></td>  <td><form:errors path="name" cssStyle="color:red"/></td>  </tr><tr>  <th>価格</th>  <td><form:input path="price"/></td>  <td><form:errors path="price" cssStyle="color:red"/></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> |

＃JSPとSpringタグライブラリの説明は、、Spring3入門の215ページ（第6章）が参考になる。

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）に下記のデータを入力して登録してみましょう。



バリデーション処理が正常にコーディングできていれば、以下のエラーが表示されるはず！

