ＳＴＥＰ０４

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検索

目次

[1 データベース接続準備 1](#_Toc428279727)

[1.1 PostgreSQLセットアップ 1](#_Toc428279729)

[1.2 DB接続用ドライバ登録 （Maven） 2](#_Toc428279730)

[1.3 DB接続設定 （Spring定義ファイル／プロパティファイル） 3](#_Toc428279731)

[2 書籍一覧Ｄａｏクラスをきちんと実装する 4](#_Toc428279732)

[2.1 書籍一覧Ｄａｏクラスの再実装 4](#_Toc428279734)

[2.2 画面表示テスト 6](#_Toc428279735)

# データベース接続準備



## PostgreSQLセットアップ

PostgreSQLデータベースは、STEP1でインストール済である。

以下の内容でDB接続できることを確認すること。

PostgreSQLの接続情報

|  |  |
| --- | --- |
| 管理ユーザ | postgres |
| パスワード | nttat123 |
| データベース | springdb |

以下の書籍テーブル（book）が存在することを確認する。

|  |  |  |
| --- | --- | --- |
| テーブル名 | | book |
| # | カラム名 | 型 | 制約 | 備考 |
| 1 | id | SERIAL | PK | SEQUENCE book\_id\_seq を用いて自動付番する |
| 2 | isbn | CHAR (17) | NOT NULL |  |
| 3 | name | VARCHAR(64) | NOT NULL |  |
| 4 | price | INTEGER | NOT NULL |  |
| 5 | lastup | TIMESTAMP |  | default CURRENT\_TIMESTAMP |

書籍テーブルを検索し、数行のサンプルデータが登録されていることを確認する。

|  |
| --- |
| SELECT \* FROM BOOK ; |

## DB接続用ドライバ登録 （Maven）

１）Maven設定ファイルにDB接続ドライバ（PostgreSQL + JDBC）を追記

|  |
| --- |
| /pom.xml |
| ● 41行目付近 |
| 41  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>${org.springframework-version}</version>  </dependency>  47  <!-- PostgreSQL + JDBC -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>${org.springframework-version}</version>  </dependency>  <dependency>  <groupId>org.apache.tomcat</groupId>  <artifactId>tomcat-jdbc</artifactId>  <version>8.0.37</version>  </dependency>  <dependency>  <groupId>org.postgresql</groupId>  <artifactId>postgresql</artifactId>  <version>9.4.1211.jre7</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（pom.xml）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～数分がかかる場合があるが、終わるまで待っているほうがよい。

## DB接続設定 （Spring定義ファイル／プロパティファイル）

１）DB接続情報を定義するプロパティファイルを新規作成＋コーディング

|  |
| --- |
| /src/main/resources/jdbc.properties （新規作成） |
| jdbc.driverClassName=org.postgresql.Driver  jdbc.url=jdbc:postgresql://localhost:5432/springdb  jdbc.username=postgres  jdbc.password=nttat123 |

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

データソースと Spring JDBCデータアクセスBean定義を追加

|  |
| --- |
| /src/main/webapp/WEB-INF/spring/application-context-biz.xml |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="  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">    <!-- Defines shared resources visible to all other web components -->  <context:component-scan base-package="jp.sample.bookmgr.biz" />    14  <!-- データソース用のリソース -->  <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  <property name="location" value="classpath:/jdbc.properties" />  </bean>    <!-- データソース設定 (for tomcat dbcp) -->  <bean id="dataSource" class="org.apache.tomcat.jdbc.pool.DataSource" destroy-method="close">  <property name="driverClassName" value="${jdbc.driverClassName}" />  <property name="url" value="${jdbc.url}" />  <property name="username" value="${jdbc.username}" />  <property name="password" value="${jdbc.password}" />  </bean>    <!-- JDBC Template Bean -->  <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">  <property name="dataSource" ref="dataSource" />  </bean>  <!-- JDBC Named Template Bean -->  <bean id="namedJdbcTemplate" class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">  <constructor-arg ref="dataSource" />  </bean>    </beans> |

# 書籍一覧Ｄａｏクラスをきちんと実装する

書籍一覧Ｄａｏは、これまで（えせ）書籍一覧を返していたが、ここで、PostgreSQLデータベースを検索して書籍一覧を取得できるよう、きちんと実装する。



## 書籍一覧Ｄａｏクラスの再実装

１）書籍一覧Ｄａｏ実装クラスのクラス名を適切なものに変更

ListBookDaoImpl を ListBookDaoSpringJdbc に変更、＜リファクタリング＞ を使用する。

ListBookDaoImpl.javaを選択して右クリック　[Refacter]-[Rename]

Rename Compilation Unitウインドウ　New Nameに変更後の名称 ListBookDaoSpringJdbc を入力してFinish

＜リファクタリングの仕方も覚えよう＞

２）書籍一覧Ｄａｏインターフェース修正、Throwする例外オブジェクトを正しく再定義する

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.biz.dao.ListBookDao.java |
| package jp.sample.bookmgr.biz.dao;  import java.util.List;  import org.springframework.dao.DataAccessException;  import jp.sample.bookmgr.biz.domain.Book;  /\*\*  \* 書籍一覧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 ListBookDao {  /\*\*  \* 書籍データベースから書籍一覧データを取得するメソッド  \* @return 書籍一覧情報  \*/  public List<Book> getBookList() throws DataAccessException;  } |

３）書籍一覧Ｄａｏ実装クラスをきちんと再コーディング

|  |
| --- |
| /src/main/java/jp.sample.bookmgr.biz.dao.ListBookDaoSpringJdbc.java |
| package jp.sample.bookmgr.biz.dao;  import java.sql.ResultSet;  import java.sql.SQLException;  // import java.util.ArrayList; → 不要となるので、この宣言は削除してください。  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.dao.DataAccessException;  import org.springframework.jdbc.core.JdbcTemplate;  import org.springframework.jdbc.core.RowMapper;  import org.springframework.stereotype.Repository;  import jp.sample.bookmgr.biz.domain.Book;  /\*\*  \* 書籍一覧データアクセス実装クラス  \* @author 長住@NTT-AT  \* @version 1.0  \*/  @Repository // リポジトリ（Dao）クラスとしてDI可能というアノテーションを宣言  public class ListBookDaoSpringJdbc implements ListBookDao {  /\*\*  \* JDBC制御クラス  \*/  @Autowired  private JdbcTemplate template;  /\*\*  \* 書籍データベースから書籍一覧データを取得するメソッド  \* @return 書籍一覧情報  \*/  @Override  public List<Book> getBookList() throws DataAccessException {    // DBマッピングインナークラスの定義  class BookRowMapper implements RowMapper<Book> {  @Override  public Book mapRow(ResultSet result, int row) throws SQLException {  Book book = new Book();  book.setId(result.getInt("ID"));  book.setIsbn(result.getString("ISBN"));  book.setName(result.getString("NAME"));  book.setPrice(result.getInt("PRICE"));  return book;  }  }  // DBマッピングクラスオブジェクト取得  RowMapper<Book> rowMapper = new BookRowMapper();    // 書籍情報検索SQL  String sql = "SELECT id, isbn, name, price FROM book ORDER BY id";    // DBを検索して書籍情報リスト取得  List<Book> bookList = template.query(sql, rowMapper);    return bookList;  }  } |

＃Spring JDBC を利用したDB接続および検索方法についての詳細は、

＃「Spring3入門」第4章が最も分かりやすい。

## 画面表示テスト

ここで、書籍一覧画面を表示して、データベースに登録された書籍情報が表示されるかどうかをチェックすること。

