

Spring 開発演習 説明資料

目次

1. 演習概要	1
1 目的	1
2 テーマ	1
3 成果物	1
2. 演習詳細	1
1 使用技術	1
3 画面作成	2
4 演習形式	2

1. 演習概要

1 目的

本演習は、Spring の単元までに学んだ講義内容の理解度、及び実装力を測ることを目的とする。

2 テーマ

架空の書店で扱っている書籍情報を閲覧できる Web アプリケーションを開発することを課題とする。実装期間として約 2 日間設ける。

3 成果物

演習における成果物は下記とする。

成果物
ソースコード
発表会資料

2. 演習詳細

1 使用技術

本システムを構築する際に用いる技術は以下のようになる。

使用技術
Java (Oracle Java SE Development Kit 16)
Spring Boot 2.7.5
Oracle Database 21c Express Edition

アプリケーションの動作確認には「Google Chrome」を使用する。

2 画面作成

画面デザインは受講生が任意で調整する。

ただし、画面中のブロック構成、入力フォームの項目数などは研修の指定により実装する。

3 演習形式

演習は個人形式となる。

受講生は、予め提示したシステム要件に沿って、既存のシステムに未実装の機能を追加実装する。「**必須機能**」は全員が必ず実装する。また、「**追加機能**」は必須機能が全て完成した後に実装するものとする。

機能一覧

必須機能	追加機能
ログイン（実装済み）	ログイン中のユーザ名表示
ログアウト（実装済み）	入力チェック
書籍一覧表示	
書籍名検索	
ジャンル名検索	

また、テーブル構成は以下の3テーブルとする。

book テーブル（書籍情報）

No	論理名称	物理名称	データ型	桁数	制約
1	書籍 ID	book_id	NUMBER	5 桁	PRIMARY KEY
2	書籍名	book_name	VARCHAR2	60 文字	NOT NULL
3	著者	author	VARCHAR2	30 文字	NOT NULL
4	発行日	publication_date	DATE	-	NOT NULL
5	在庫	stock	NUMBER	4 桁	NOT NULL
6	ジャンル ID	genre_id	NUMBER	2 桁	NOT NULL FOREIGN KEY

genre テーブル（ジャンル情報）

No	論理名称	物理名称	データ型	桁数	制約
1	ジャンル ID	genre_id	NUMBER	2 桁	PRIMARY KEY
2	ジャンル名	genre_name	VARCHAR2	30 文字	NOT NULL

book_user テーブル（ユーザ情報）

No	論理名称	物理名称	データ型	桁数	制約
1	ユーザ ID	book_user_id	NUMBER	5 桁	PRIMARY KEY
2	ユーザ名	book_user_name	VARCHAR2	30 文字	NOT NULL
3	パスワード	password	VARCHAR2	16 文字	NOT NULL

3. 演習の環境構築

1 SQL 文の実行

LMS で配布する「[PRGDE004]Spring_開発演習_SQL 文_1.1.txt」を SQLDeveloper または sqlplus で実行し、ユーザ・テーブル・レコードを作成、挿入する。

2 編集用プロジェクトのインポート

LMS で配布する「[PRGDE004]Spring_開発演習_ソースコード(編集用)_1.3.zip」をダウンロードし Eclipse にインポートする。

【インポート手順】

- ・上部タブの ファイル > インポート を選択
- ・一般 > 既存プロジェクトをワークスペースへ を選択
- ・ラジオボタン：アーカイブファイルの選択 を選択し、右部の「参照」を押下
- ・ダウンロードした「[PRGDE004]Spring_開発演習_ソースコード(編集用)_1.3.zip」を選択
- ・プロジェクトのボックス内に「book_list」が表示されるので選択し、「完了」を押下

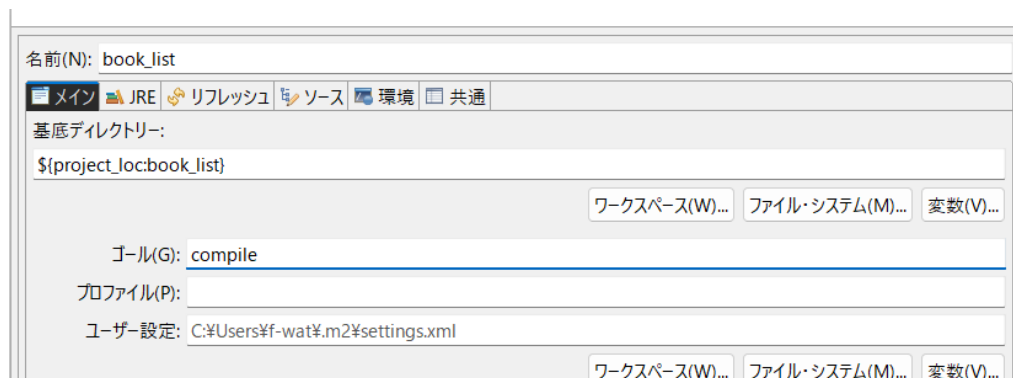
インポート後はプロジェクトを起動し[localhost:2222/book_list/]にブラウザでアクセスしログイン画面が表示されることを確認する。

※Q&A プロジェクトがビルドしない場合

Q プロジェクトを実行すると ClassNotFoundException が発生する

A Java ファイルのコンパイルがされていない。プロジェクトを右クリック→「**実行→9 Maven ビルド**」を選択しゴールに以下のコマンドライン引数を貼り付けて実行する。

compile



Q プロジェクトを実行すると「XXX has been compiled by a more recent version of the Java Runtime～」エラー（UnsupportedClassVersionError）と表示される

A JRE のバージョンがプロジェクトと Eclipse の実行環境とで異なっている。以下の表示でバージョンがっているか、確認をする。

① プロジェクトを右クリックし「プロパティ→サイドバー：Java コンパイラー」を選択し、

コンパイラ準拠レベルのバージョンを確認

② pom.xml の java.version タグを確認

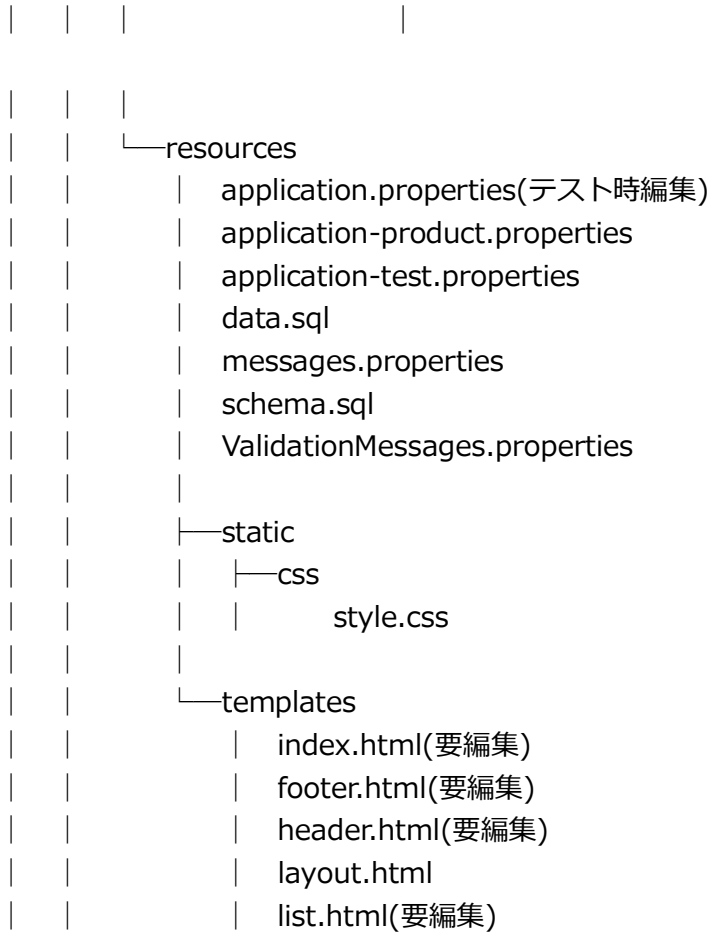
→①②が演習のバージョンとずれている場合、開発演習のバージョンに合わせ修正を行う。

3 ファイル構成・実装対象ファイル

- ・ファイル構成は以下とする。ファイル名のみのものは実装済みである。
- ・テスト結果に影響が出るため、**既存の HTML のタグ、属性などの構成は変えないこと**。また、新規で作成するファイルは既存コードをよく参照し作成すること。
- ・URL 設計は自身で考えること。ただし、既存のソースに記載のある URL はそのまま使用する。
例：書籍名検索 [/book_list/book_name]
- ・要件定義書、基本設計書、詳細設計書、テスト仕様書は自身で読み込みを行い、疑問点は適宜質問して解決すること。
- ・開発時の環境は「application-product.properties」に記載されている。
- ・テスト時の環境は「application-test.properties」に記載されている。**なお、H2DB は組み込み DB のため、SpringBoot を再起動するたびにテーブルが初期化される。**

【ファイル構成】

```
{project}
├──src
│   ├──main
│   │   ├──java
│   │   │   ├──jp
│   │   │   │   ├──co
│   │   │   │   │   ├──sss
│   │   │   │   │   │   ├──book
│   │   │   │   │   │   │   ├──BookListApplication.java
│   │   │   │   │   │   │   ├──controller
│   │   │   │   │   │   │   │   ├──SessionController.java(要編集)
│   │   │   │   │   │   │   │   ├──BookController.java(新規)
│   │   │   │   │   │   │   ├──entity
│   │   │   │   │   │   │   │   ├──BookUser.java
│   │   │   │   │   │   │   │   ├──Book.java(新規)
│   │   │   │   │   │   │   │   ├──Genre.java(新規)
│   │   │   │   │   │   │   ├──form
│   │   │   │   │   │   │   │   ├──LoginForm.java(要編集)
│   │   │   │   │   │   │   ├──repository
│   │   │   │   │   │   │   │   ├──BookUserRepository.java
│   │   │   │   │   │   │   │   ├──BookRepository.java(新規)
```



4. テストの実行と採点

1 採点基準

各機能の完成は[src/test/java]以下にあるテストコードの結果によって採点する。

例 書籍一覧表示機能 > jp.co.sss.book.test03.BookListTest.java

2 機能別テストの実行

機能が完成するごとにテストコードを実行し確認をする。

【進行例】

機能を実装→機能テスト NG→バグ修正→機能テスト OK→次の機能を実装

機能ごとのテストコードの実行は以下の手順で実施する。

1. テスト実行用のドライバファイル「chromedriver.exe」を配置する。(別紙「**web アプリケーション開発演習_chrome driver 更新手順_1.0.0.pdf**」の手順を参考にし、ドライバファイルをプロジェクトに配置する。ドライバの場所は「{project}/driver/」となる。

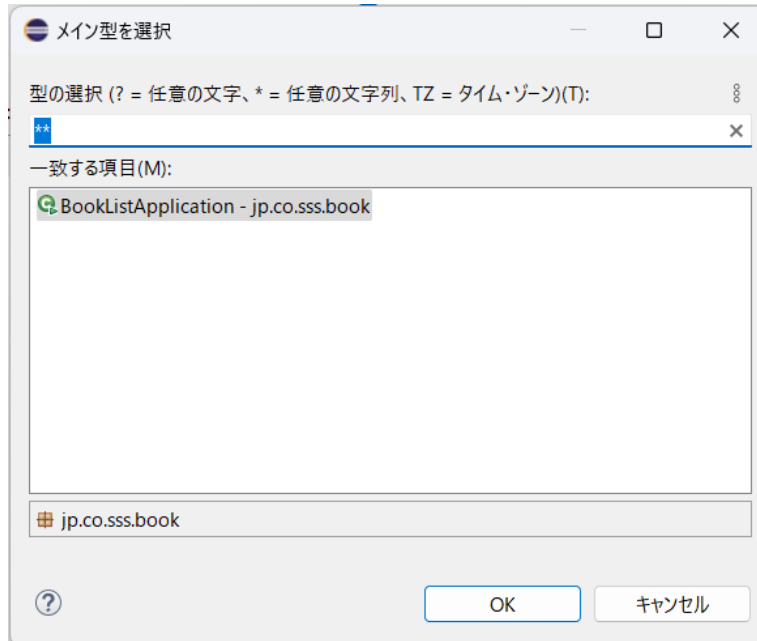
※注意

「chromedriver.exe」が配置されていない場合、テストの実行に失敗します。
テストを実行する前は必ず上記の「1.」の手順を確認、実施してください。

2. application.properties を修正しテスト用プロファイルに切り替える

```
application.properties × application-test.properties
1#開発用 テストを行う際は下記をコメントアウトする
2#spring.profiles.active=product
3
4#テスト用 テストを行う際は下記コメントを外す
5spring.profiles.active=test
```

3. SpringBoot を起動する。起動する際に選択を求められる場合は「BookListApplication」を選択する。

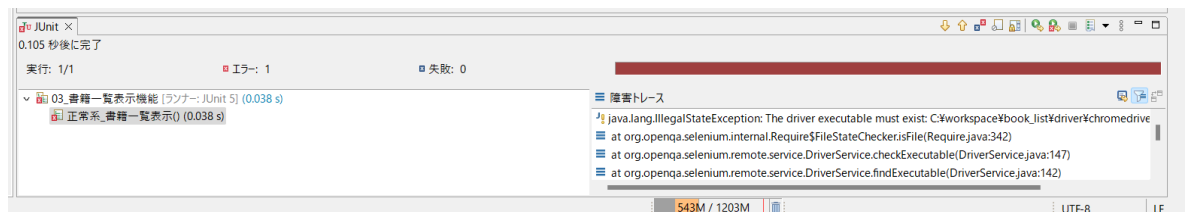


4. 該当のテストクラスを右クリックし、「実行 > 2 JUnit テスト」を選択する

実行結果がすべて OK であれば機能完成とする。

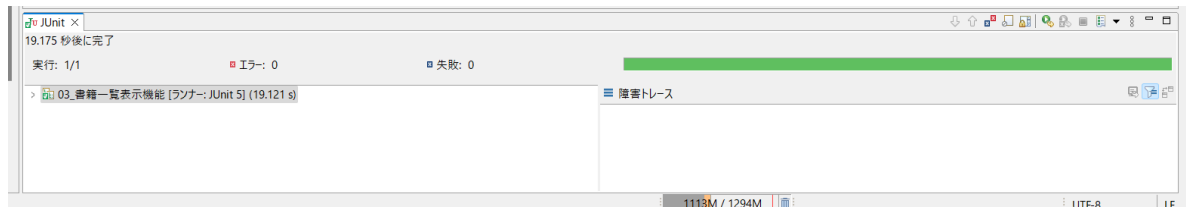
実行結果がエラーや失敗であればソースコードに間違いがあるため、デバッグを行う。

<NG の場合>



<OK の場合>

※デバッグ後は SpringBoot を一度停止し、1 の手順から再度テストを行う。



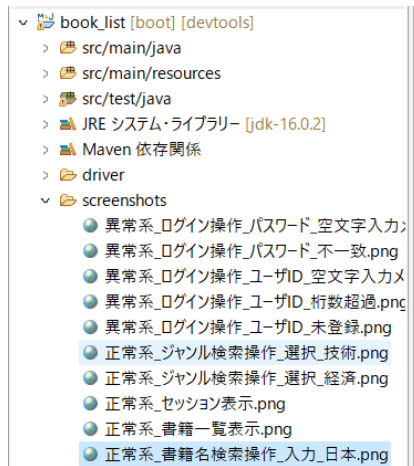
※テスト OK の後は application.properties を修正し、開発用プロファイルに戻す

3 デバッグのポイント

テストを行い要求される品質を満たすことは非常に重要である。テスト後のエビデンスの確認、障害トレースの例を以下に示す。参考にしてデバッグを行うこと。

1. エビデンスの確認

{project}/screenshots/ フォルダの中にテスト実行時のエビデンス画像が格納される

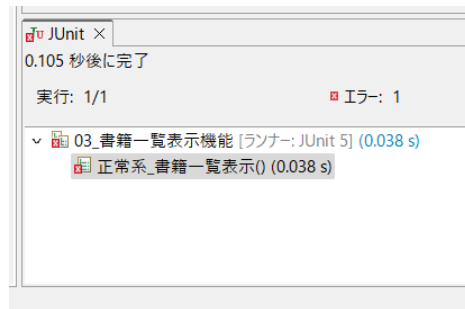


エビデンス画像が想定される画面となっているか確認する

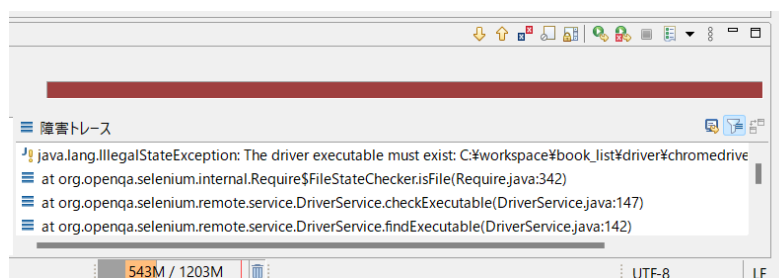
エビデンス画像がない場合は2の障害トレースを参照する

2. Junit の障害トレース

Junit ビューにある「テストクラス > テストメソッド」を1クリックすると右部に「障害トレース」が表示され、どんなエラーが発生したのか？確認することができる。



↓ 1 クリック後



また、テストメソッドをダブルクリックするとエラーが発生したテストの箇所を開くことができる。(下の画像では BookListTest.java の 54 行目にエラーが発生したことがわかる)



3. 主なエラーと対処

【原則】 デバッグする際にテストクラスのコードを修正してはならない。やむを得ず修正する際は必ず講師の許可を得ること。

- ① 実行結果がエラー かつ NoSuchElementException が発生している

原因：HTML タグや class 属性などが想定通りになっていない可能性がある。既存コードのコメントをよく参照し修正すること。

②実行結果が失敗 かつ AssertionErrorが発生している

原因：ブラウザに表示される項目（Element）が想定と違う場合に発生する。障害トレースには[**expected:<A> but was:**]という記載がされる。A が期待される値、B が実際の値である。設計書を参照し正しい表示に修正する。以下の点も併せて確認すること。


- ・ スコープに登録されている値が想定通りかどうか？
- ・ HTML に表示される表記が正しいかどうか？
- ・ HTML に表示されるフォーマット（形式）が正しいかどうか？
- ・ 表示される項目の順序が想定通りかどうか？

4 結合テストの実行と成果物提出

テストフェーズ期間に入ったら結合試験を実施する。試験は Junit および Selenium を用いての自動テストを実施する。テスト要件は「[PRGDE004]Spring_開発演習_テスト仕様書_1.1.xls」を参照すること

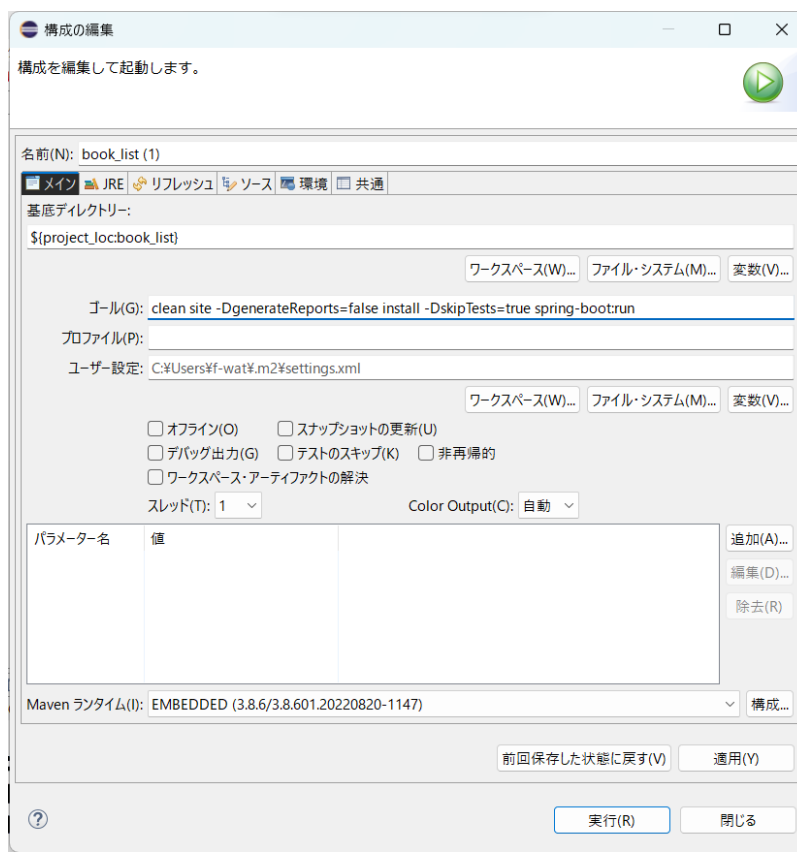
結合試験実施ならびに試験結果の提出は以下の通り行う。

1. application.properties を修正しテスト用プロファイルに切り替える



```
application.properties × application-test.properties
1#開発用 テストを行う際は下記をコメントアウトする
2#spring.profiles.active=product
3
4#テスト用 テストを行う際は下記コメントを外す
5spring.profiles.active=test
```

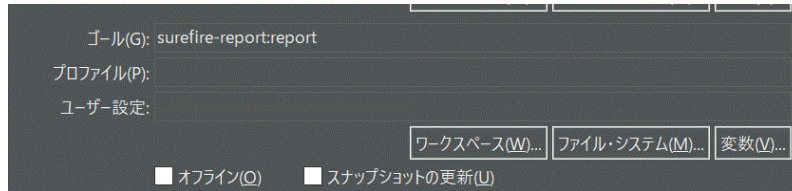
2. プロジェクトを右クリック→実行→Maven ビルド を選択しゴールに以下のコマンドライ



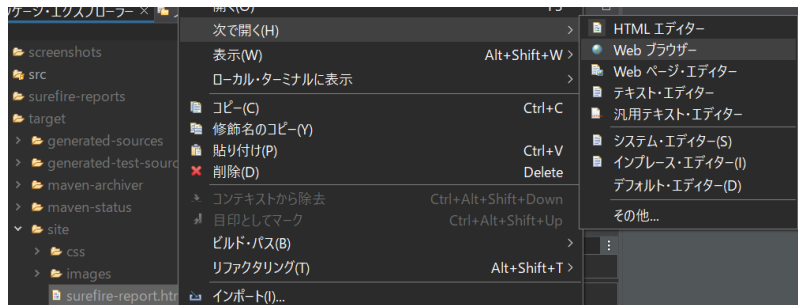
ン引数を貼り付ける（レポートのテンプレートがインストールされ、プロジェクトが起動する）
clean site -DgenerateReports=false install -DskipTests=true spring-boot:run

3. 同様に Maven ビルドを選択し以下のコマンドを貼り付ける。テストが実行されレポートが出力される。※テストの実行には十数分程度時間がかかる場合があります。Eclipse のコンソールで「BUILD SUCCESS」の文字が確認できたら、実行完了になります。

surefire-report:report



4. {project}/target/site/surefire-report.html をブラウザで開くとテスト結果が見られる。ファイルを開く際は「surefire-report.html を右クリック > 次で開く > WEB ブラウザ」を選択する。



（ファイルが見えない場合は F5 キーなどでリフレッシュすること）

※再度テストする際は 1 の項目から実施しなすこと

surefire-report.html は以下のような情報が記載されている。

- ・テスト項目（1 機能 1 パッケージ）
- ・テスト項目数
- ・実行結果（エラー、失敗）
- ・成功率

※成功率 100%の機能を完成として採点する

Package List

[Summary] [Package List] [Test Cases]

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
jp.co.sss.book.test04	3	0	0	0	100%	35.881
jp.co.sss.book.test05	2	0	0	0	100%	34.708
jp.co.sss.book.test02	1	0	0	0	100%	17.359
jp.co.sss.book.test03	1	0	0	0	100%	17.405
jp.co.sss.book.test00	1	0	0	0	100%	17.844
jp.co.sss.book.test01	1	0	0	0	100%	17.28
jp.co.sss.book	1	0	0	0	100%	3.675
jp.co.sss.book.test06	5	0	0	0	100%	69.883
jp.co.sss.book.test07	2	0	0	0	100%	34.707

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

5. {project}/target/site フォルダをコピーし、任意の場所にペーストしてから ZIP 化してサポーターに提出する。

(また、LMS にも成果物としてアップロードすること)

成果物名：書籍閲覧システム_結合試験結果_{NAME}.zip

※{NAME}は受講生名

※担当者様への共有

LMS マニュアル# 5. ファイルアップロード機能 を参照する

※講師への共有

Slack または LMS の成果物アップロード機能を用いる