

Spring Framework 【Spring Boot】 練習問題

1	はじめに	3
2	Lesson01_01	5
3	Lesson01_02	6
4	Lesson01_03	7
5	Lesson01_04	8
6	Lesson01_05	9
7	Lesson01_06	10
8	Lesson01_07	11
9	Lesson01_08	13
10	Lesson01_09	15
11	Lesson01_10	16
12	Lesson02_01	17
13	Lesson02_02	19
14	Lesson02_03	22
15	Lesson02_04	25
16	Lesson02_05	26
17	Lesson02_06	28
18	Lesson02_07	30
19	Lesson02_08	32
20	Lesson02_09	34
21	Lesson02_10	36
22	Lesson02_11	39
23	Lesson02_12	40
24	Lesson03_01	42
25	Lesson03_02	45
26	Lesson04_01	46
27	Lesson04_02	48
28	Lesson04_03	50
29	Lesson05_01	52
30	Lesson05_02	54
31	Lesson06_01	56
32	Lesson06_02	57
33	Lesson07_01	58
34	Lesson07_02	60
35	Lesson07_03	61
36	Lesson07_04	62
37	Lesson07_05	64

38	Lesson08_01	66
39	Lesson08_02	68

1 はじめに

はじめに、下記の内容を確認して練習問題を始めてください。

1. プロジェクトの作成

Eclipse 画面から[ファイル]>[新規]>[Spring スターター・プロジェクト]を選択してプロジェクトを作成します。プロジェクトは次の内容で作成してください。

名前： spring_training
グループ： jp.co.sss.training
成果物： spring_training
パッケージ： jp.co.sss.training

なお、「新規 Spring スターター・プロジェクトし依存関係」画面にて以下の項目を選択して下さい。

- Spring Web
- Spring Boot DevTool
- Thymeleaf
- Validation

2. コンテキストパスの設定

コンテキストパスは「/training」と設定してください。

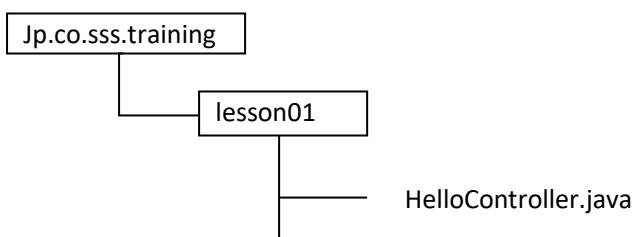
3. ポート番号の設定

プロジェクトのポート番号には任意の番号を設定してください。
他のアプリケーションと重複しないようにご注意ください。

4. パッケージの作成

問題ごとにパッケージを作成します。

例えば、演習問題 1 を解くときは、



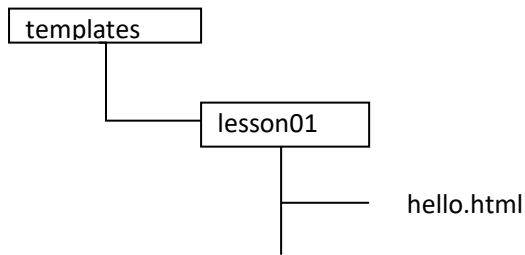
このように作ります。

※一部クラスは専用の各問題共通のパッケージを作成する場合があります。

5. フォルダの作成

html ファイルを入れるフォルダも問題ごとに作ります。

例えば、演習問題 1 を解くときは、



このように作ります。

6. 問題の説明文について

各問題の説明文には、既に存在するファイルに設定を追記する「追記事項」欄や、テーブル作成やデータの登録を行う「実行する SQL」欄の記載をしている場合があります。その際には、予めその内容に従って各設定を行った上で問題作成に取り組んでください。「テキスト参考箇所」の項目がある問題は、テキスト上で参考となる該当ページを記載しています。「ヒント」の項目には、問題に取り組むにあたってのヒントを記載しているので、迷った場合に参考にしてください。

※※ 注意！！※※

クラスやインターフェイスはパッケージが別であっても、必ず別の名前で作成してください。他のパッケージのクラスやインターフェイスと同じ名前を付けるとアプリケーションの起動に失敗する場合があります。html ファイルについては、フォルダがわかれていれば同名のファイルでも問題ありません。

2 Lesson01_01

コントローラとビューを用いて、画面に「こんにちは」と表示してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_01

クラス名...Hello0101Controller

ビュー：

フォルダ名...lesson01_01

ファイル名...hello.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_01/hello」

実行例：



テキスト参考箇所：第 2 章 基本操作 P57 ～ P67

3 Lesson01_02

「フォワード」という名前のリンクがある画面を表示してください。

リンクをクリックしたらフォワードによる画面遷移を行い、遷移先の画面に「遷移先のページ」と表示してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_02

クラス名...Input0102Controller

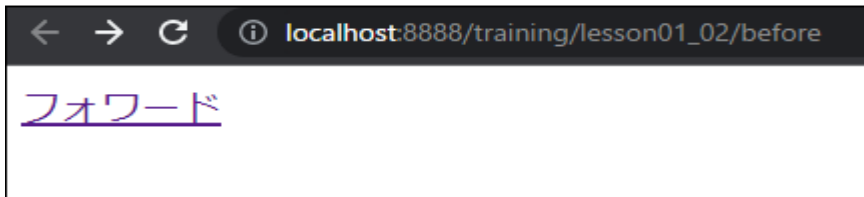
ビュー：

フォルダ名...lesson01_02

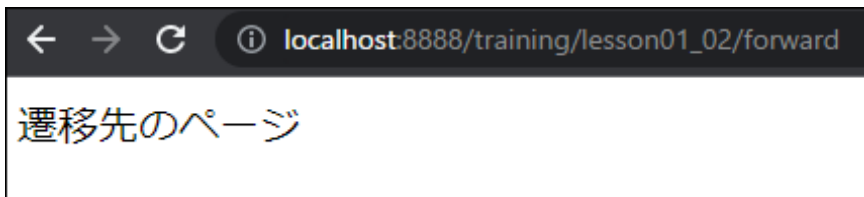
ファイル名...before.html、after.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_02/before」

実行例



リンクをクリック後



テキスト参考箇所：第3章 画面遷移 P73 ～ P87

4 Lesson01_03

「リダイレクト」という名前のリンクがある画面を表示してください。「リダイレクト」リンクをクリックした場合に、リダイレクトによる画面遷移を行ってください。遷移先の画面には、「遷移先のページ」と表示してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_03

クラス名...Input0103Controller

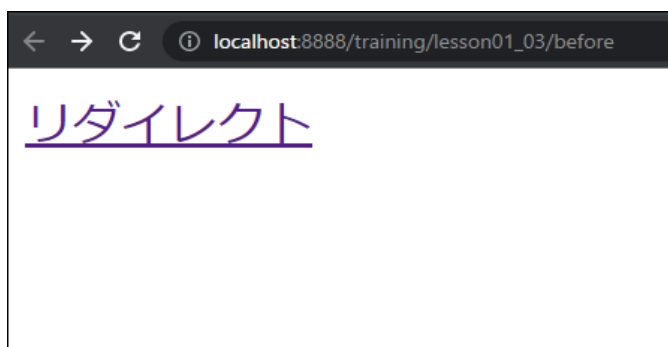
ビュー：

フォルダ名...lesson01_03

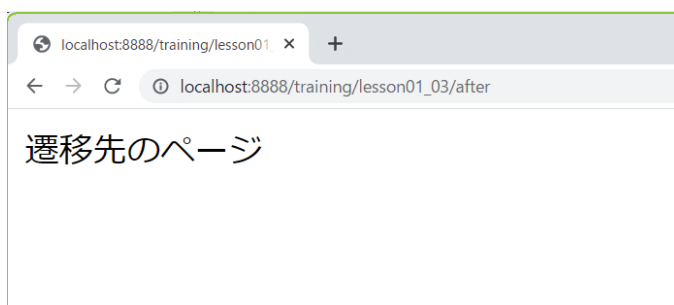
ファイル名...before.html、after.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_03/before」

リダイレクトの実行例



リダイレクトをクリック後



テキスト参考箇所：第 3 章 画面遷移 P90 ～ P95

5 Lesson01_04

コントローラに、メソッド「hello()」と「world()」を定義してください。

「hello()」メソッドで「http://localhost:8888/training/lesson01_04/world」へリダイレクトを実行し「world()」メソッドを実行してください。「world()」メソッドでは、「world.html」に画面遷移を行ってください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_04

クラス名...Hello0104Controller

ビュー：

フォルダ名...lesson01_04

ファイル名...world.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_04/hello」

実行例：

リダイレクト後の URL 「http://localhost:8888/training/lesson01_04/world」



テキスト参考箇所：第 3 章 画面遷移 P90 ～ P95

6 Lesson01_05

任意の文字列を入力するフォームを表示してください。

「送信」ボタンが押されたときに入力された文字列を GET 送信し、Eclipse の「コンソール上」に出力してください。※出力以外にプログラム内で文字列に対して処理をする必要はありません。また、出力処理を行ったコントローラでは、元の「getForm.html」に再び遷移する形としてください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_05

クラス名...Input0105Controller

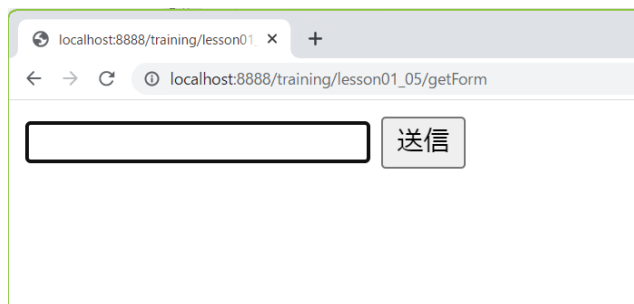
ビュー：

フォルダ名...lesson01_05

ファイル名...getForm.html

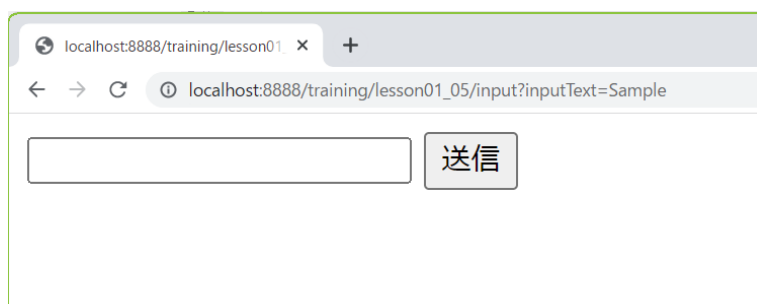
※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_05/getForm」

実行例



「Sample」と入力し、「送信」ボタンをクリック後

URL : 「http://localhost:8888/training/lesson01_05/input?inputText=Sample」



テキスト参考箇所：第 4 章 フォーム処理 P96 ～ P106

7 Lesson01_06

任意の文字列を入力するフォームを表示してください。

「送信」ボタンが押されたとき、入力された文字列を POST 送信し、Eclipse の「コンソール上」に出力してください。※出力以外にプログラム内で文字列に対して処理をする必要はありません。また、出力処理を行ったコントローラでは、元の「postForm.html」に再び遷移する形としてください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_06

クラス名...Input0106Controller

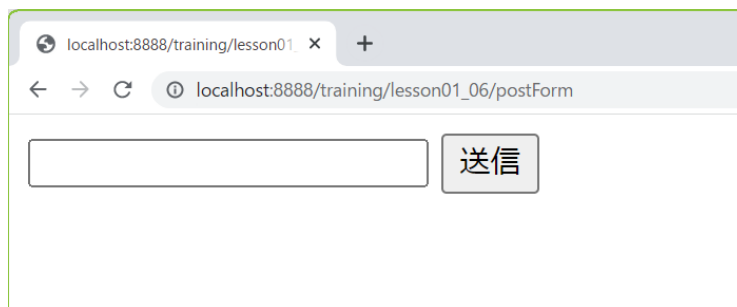
ビュー：

フォルダ名...lesson01_06

ファイル名...postForm.html

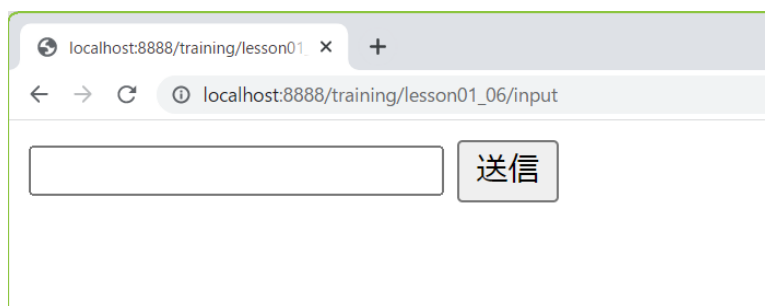
※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_06/postForm」

実行例



「Sample」と入力し、「送信」ボタンをクリック後

URL：「http://localhost:8888/training/lesson01_06/input」



テキスト参考箇所：第4章 フォーム処理 P107 ～ P112

8 Lesson01_07

ログイン画面で、ユーザーID とパスワード（ともに文字列）を入力するフォームを表示してください。「ログイン」ボタンが押されたとき、入力されたユーザーID とパスワードの文字列が一致していれば、top.html に画面遷移するメソッドにリダイレクトしてください。それ以外の場合は、元のログイン画面を再表示してください。top.html には「戻る」リンクを作成し、押下時にログイン画面に遷移するようにしてください。

ログインボタンが押されたときの処理を行うメソッドは、次の形式で定義すること（パラメータを String 型の引数で受け取る）。

```
@RequestMapping(path = "/lesson01_07/login", method = RequestMethod.POST)
public String doLogin(String userId, String password) {
```

入力されたユーザーID とパスワードの文字列が一致したときにリダイレクト先のメソッドは、次の形式で定義すること

```
@RequestMapping(path = "/lesson01_07/top")
public String top() {
```

なお、ユーザーID に何も入力せずに「ログイン」ボタンが押された場合も、元のログイン画面を再表示してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_07

クラス名...Login0107Controller

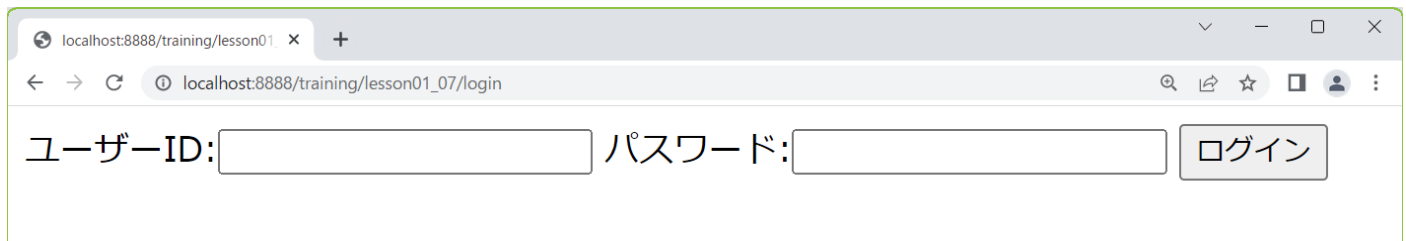
ビュー：

フォルダ名...lesson01_07

ファイル名...login.html、top.html

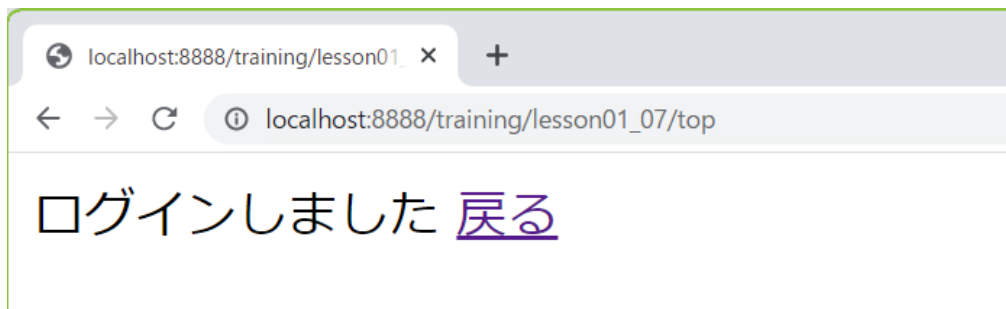
※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_07/login」

実行例：

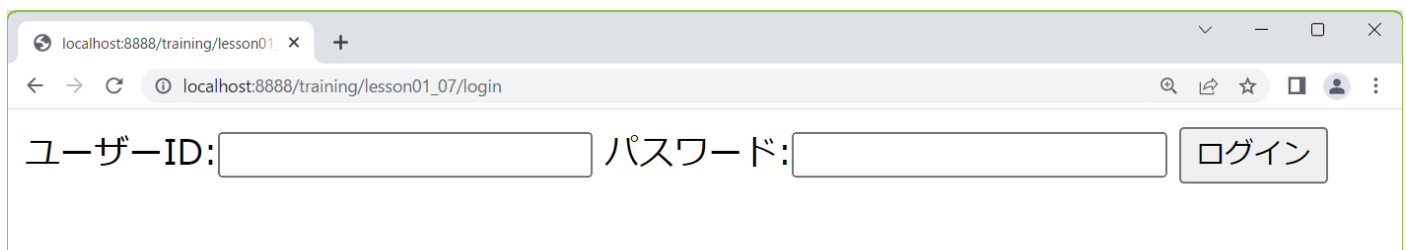


The screenshot shows a web browser window with the address bar displaying 'localhost:8888/training/lesson01_07/login'. The page content includes a form with two input fields: 'ユーザーID:' followed by a text box, and 'パスワード:' followed by a text box. To the right of these fields is a button labeled 'ログイン'.

ユーザーID、パスワードともに同じ文字列を入力した場合



top.html の「戻る」リンクをクリックした後



テキスト参考箇所：第4章 フォーム処理 P96 ～ P113

9 Lesson01_08

ログイン画面で、ユーザーID とパスワード（ともに文字列）を入力するフォームを表示してください。「ログイン」ボタンが押されたとき、入力されたユーザーID とパスワードの文字列が一致していれば、top.html に画面遷移するメソッドにリダイレクトしてください。それ以外の場合は、元のログイン画面を再表示してください。top.html には「戻る」リンクを作成し、ログイン画面に遷移するようにしてください。

ただし、ログインボタンが押されたときの処理を行うメソッドは、次の形式で定義すること（入力したパラメータを LoginForm クラスの引数で受け取る）。

```
@RequestMapping(path = "/lesson01_08/login", method = RequestMethod.POST)
public String doLogin(LoginForm loginForm) { ↓
```

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_08

クラス名...Login0108Controller

フォーム：

パッケージ名...jp.co.sss.training.form

クラス名...LoginForm

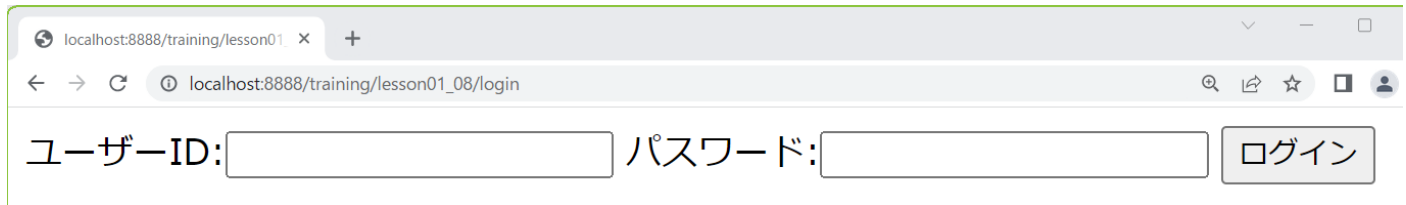
ビュー：

フォルダ名...lesson01_08

ファイル名...login.html、top.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_08/login」

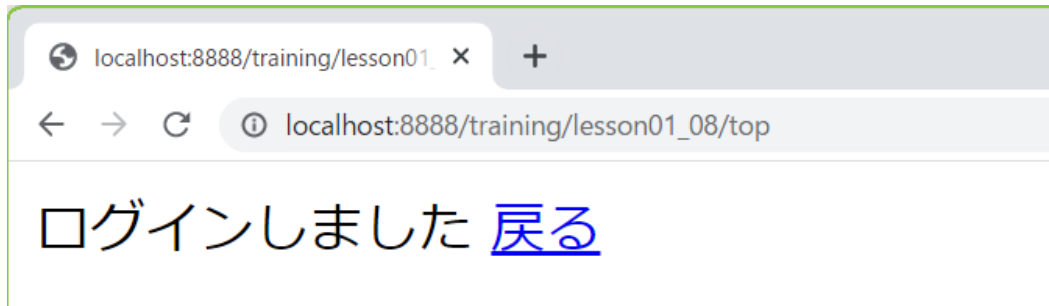
実行例



localhost:8888/training/lesson01_08/login

ユーザーID: パスワード: ログイン

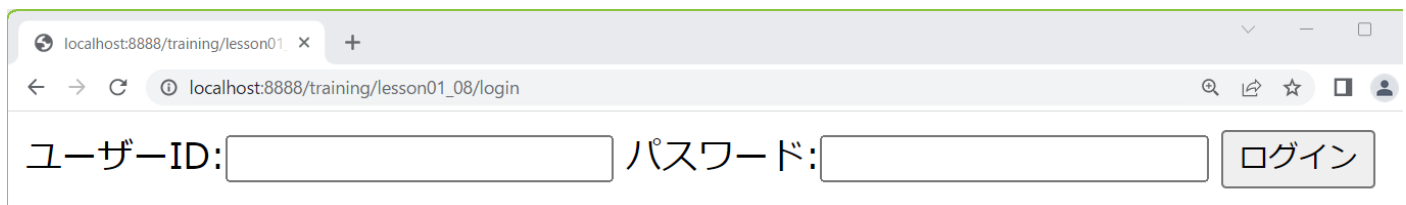
ユーザーID、パスワードともに同じ文字列を入力した場合



localhost:8888/training/lesson01_08/top

ログインしました [戻る](#)

top.html の「戻る」リンクをクリックした後



localhost:8888/training/lesson01_08/login

ユーザーID: パスワード: ログイン

テキスト参考箇所：第4章 フォーム処理 P116 ～ P122

ヒント：

- LoginForm クラスは下記のフィールドを定義して作成します。

```
public class LoginForm {  
    private String userId;  
    private String password;  
}
```

10 Lesson01_09

Input.html で入力した内容をリクエストスコープに登録し、その内容を画面遷移先の result.html で表示してください。表示形式は実行例に従ってください。

※入力した内容（パラメータ）をフォームで受け取るかどうかは任意とします。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_09

クラス名...Request0109Controller

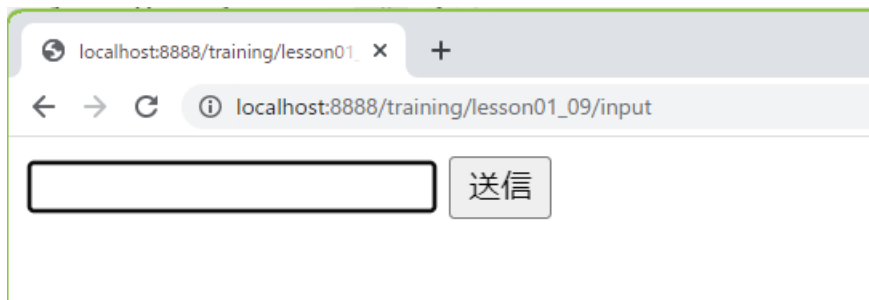
ビュー：

フォルダ名...lesson01_09

ファイル名...input.html、result.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_09/input」

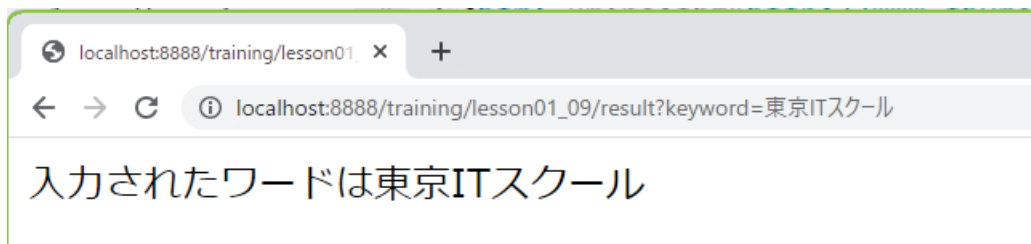
実行例



localhost:8888/training/lesson01_09/input

送信

「東京 IT スクール」と入力し、「送信」ボタンをクリック後



localhost:8888/training/lesson01_09/result?keyword=東京ITスクール

入力されたワードは東京ITスクール

テキスト参考箇所：第 5 章 スコープ P123 ～ P132

11 Lesson01_10

ログイン画面で、ユーザーID とパスワード（ともに文字列）を入力するフォームを表示してください。「ログイン」ボタンが押されたとき、入力されたユーザーID とパスワードの文字列が一致していれば、セッション（HttpSession）にユーザーID を記録し、遷移後の画面にそのユーザーID を表示してください。また、遷移後の画面の「ログアウト」リンクが押されたときは、セッション情報を破棄した上で再度ログイン画面を表示してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson01_10

クラス名...Login0110Controller

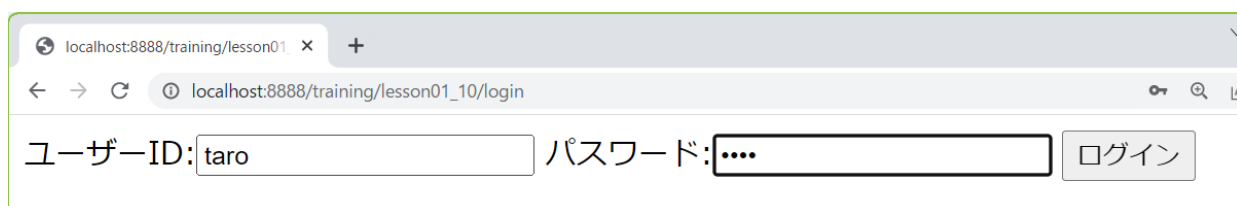
ビュー：

フォルダ名...lesson01_10

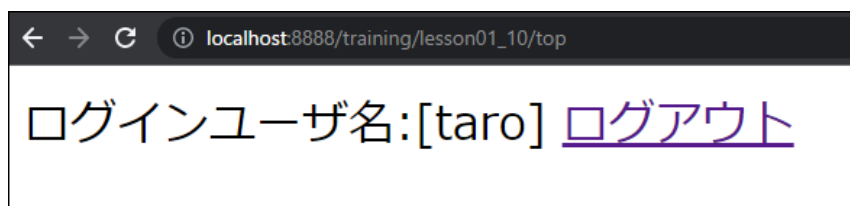
ファイル名...login.html（※ログイン画面）、top.html（※遷移後の画面）

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_10/login」

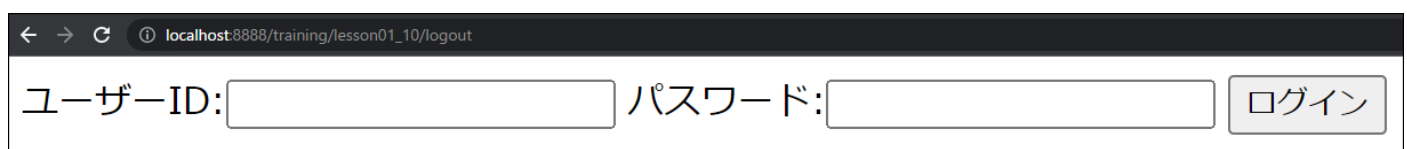
実行例（login.html）



ログイン成功時（top.html）



top.html の「ログアウト」リンクをクリックした後（login.html）



テキスト参考箇所：第 5 章 スコープ P133 ～ P141

12 Lesson02_01

training_user テーブルを作成して、データベースに登録されたユーザー情報を一覧表示する画面を作成してください。（表示内容は実行例を参考にしてください。）

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson02_01

クラス名...User0201Controller

ビュー：

フォルダ名...lesson02_01

ファイル名...index.html

エンティティ：

パッケージ名...jp.co.sss.training.entity

クラス名...TrainingUser

リポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...TrainingUserRepository

利用するテーブル：

テーブル名...training_user テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_01/index」

追記事項：

「application.properties」に、以下の内容を追記してください。

```
spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/xepdb1 ↓
spring.datasource.username=spring_user ↓
spring.datasource.password=systemsss
```

実行する SQL :

下記の SQL を実行し training_user テーブルを作成してください。

※実行前に、「application.properties」の内容を参考に DB 接続用ユーザーを新しく作成してください。

```
CREATE TABLE training_user(  
  id NUMBER(4) PRIMARY KEY,  
  user_id VARCHAR2(10) NOT NULL,  
  password VARCHAR2(10) NOT NULL  
);  
  
CREATE SEQUENCE seq_training_user NOCACHE;  
INSERT INTO training_user VALUES (seq_training_user.NEXTVAL,'kadokura','pass01');  
INSERT INTO training_user VALUES (seq_training_user.NEXTVAL,'yamamoto','pass02');  
INSERT INTO training_user VALUES (seq_training_user.NEXTVAL,'ogawa','pass03');  
INSERT INTO training_user VALUES (seq_training_user.NEXTVAL,'kuramae','pass04');  
INSERT INTO training_user VALUES (seq_training_user.NEXTVAL,'yamashita','pass05');  
INSERT INTO training_user VALUES (seq_training_user.NEXTVAL,'kanagawa','pass06');  
  
commit;
```

実行例



The screenshot shows a web browser window with the address bar displaying 'localhost:8888/training/lesson02_01/index'. The main content area contains a table with two columns: '番号' (Number) and 'ユーザーID' (User ID). The table lists six users with their corresponding IDs and numbers.

番号	ユーザーID
1	kadokura
2	yamamoto
3	ogawa
4	kuramae
5	yamashita
6	kanagawa

テキスト参考箇所 : 第 6 章 データベース操作(CRUD) P149 ~ P168

13 Lesson02_02

データベースに商品（商品 ID、商品名、価格、ジャンル ID）データが登録されている。価格の安い順に並び替えを行い、結果を実行例のように表形式で出力してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson02_02

クラス名...Item0202Controller

エンティティ：

パッケージ名...jp.co.sss.training.entity

クラス名...Item

リポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...ItemRepository

ビュー：

フォルダ名...lesson02_02

ファイル名...index.html

利用するテーブル：

テーブル名...item テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_02/index」

実行する SQL :

- 下記の SQL で item テーブルを作成します。

```
CREATE TABLE item (  
  id NUMBER(3) PRIMARY KEY,  
  name VARCHAR2(100),  
  price NUMBER(5),  
  genre_id NUMBER(3)  
);  
  
create SEQUENCE seq_item NOCACHE;  
  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'りんご', 100, 1);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'みかん', 200, 1);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'いちご', 300, 1);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'バナナ', 100, 1);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'ぶどう', 200, 1);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'なし', 300, 1);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'やさしい Java', 4000, 2);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'やさしい Spring', 3000, 2);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'やさしい MySQL', 4000, 2);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'やさしい Seasar2', 3000, 2);  
INSERT INTO item VALUES (seq_item.NEXTVAL, 'やさしい Oracle', 4000, 2);  
  
commit;
```

実行例：

ID	商品名	価格	ジャンルID
1	りんご	100	1
4	バナナ	100	1
2	みかん	200	1
5	ぶどう	200	1
6	なし	300	1
3	いちご	300	1
8	やさしいSpring	3000	2
10	やさしいSeasar2	3000	2
7	やさしいJava	4000	2
9	やさしいMySQL	4000	2
11	やさしいOracle	4000	2

テキスト参考箇所：第 6 章 データベース操作(CRUD) P170 ～ P174

ヒント：

昇順に検索するメソッドは

`findOrderBy[エンティティのフィールド名]Asc()` と定義します。

14 Lesson02_03

Lesson02_01 で作成した training_user テーブルを利用して、Lesson02_01 と同様の一覧画面を新しく作成してください。更に、ユーザー名をクリックしたときに、詳細情報画面（クリックされたユーザーの情報を 1 件だけ表示する画面）に遷移するようにしてください。

なお、データベースから取得したユーザーの情報は、コントローラの中で「**BeanUtils.copyProperties()**」メソッドを使用して Bean クラスのオブジェクトにコピーした上で、詳細画面に遷移してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson02_03

クラス名...User0203Controller

ビュー：

フォルダ名...lesson02_03

ファイル名...index.html、show.html

Bean：

パッケージ名...jp.co.sss.training.bean

クラス名...UserBean

エンティティ：

パッケージ名...jp.co.sss.training.enntity

クラス名...TrainingUser

リポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...TrainingUserRepository

利用するテーブル：

テーブル名...training_user テーブル

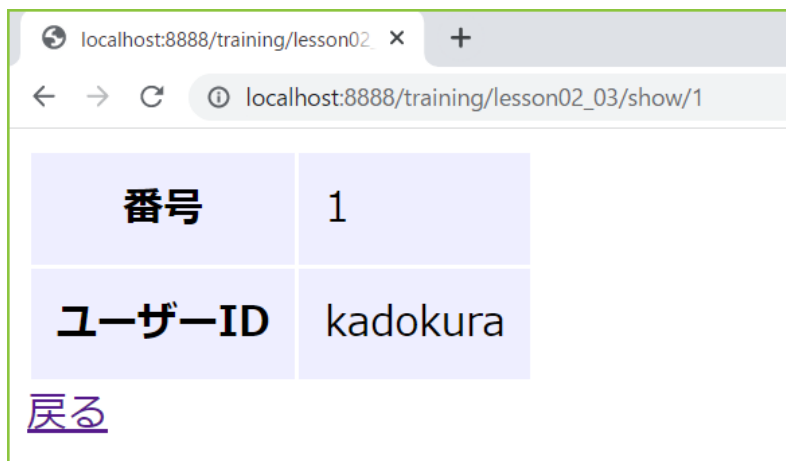
※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_03/index」

実行例：



番号	ユーザーID
1	kadokura
2	yamamoto
3	ogawa
4	kuramae
5	yamashita
6	kanagawa

ユーザーID をクリック（番号 1 のユーザーをクリックした場合）



番号	1
ユーザーID	kadokura

[戻る](#)

テキスト参考箇所：第 6 章 データベース操作(CRUD) P161 ～ P169, P175～P186

ヒント:

UserBean クラスは下記の通り作成します。

```
public class UserBean implements Serializable {  
    private Long id;  
    private String userId;  
    private String password;  
  
    public Long getId() {  
        return id;  
    }  
  
    public void setId(Long id) {  
        this.id = id;  
    }  
  
    public String getUserId() {  
        return userId;  
    }  
  
    public void setUserId(String userId) {  
        this.userId = userId;  
    }  
  
    public String getPassword() {  
        return password;  
    }  
  
    public void setPassword(String password) {  
        this.password = password;  
    }  
}
```

「BeanUtils.copyProperties()」メソッドの引数は第1引数にコピー元、第2引数にコピー先を指定してください。

15 Lesson02_04

Lesson02_02 で作成した item テーブルを利用して、「価格が 200 円、かつ、ジャンル ID が 1」である商品を検索し、結果を実行例のような表形式で出力してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ :

パッケージ名...jp.co.sss.training.lesson02_04

クラス名...Item0204Controller

リポジトリ :

パッケージ名...jp.co.sss.training.repository

インターフェイス名...ItemRepository

ビュー :

フォルダ名...lesson02_04

ファイル名...index.html

利用するテーブル :

テーブル名...item テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_04/index」

実行例 :

ID	商品名	価格	ジャンルID
2	みかん	200	1
5	ぶどう	200	1

テキスト参考箇所 : 第 6 章 データベース操作(CRUD) P193 ~ P196

ヒント :

AND 検索を条件とするメソッドは、

findBy[エンティティのフィールド名 1]And[エンティティのフィールド名 2]([検索条件の値 1], [検索条件の値 2])

と定義します。

16 Lesson02_05

Lesson02_02 で作成した item テーブルを利用して、「価格が 200 円、または、ジャンル ID が 2」である商品を検索し、結果を表形式で出力してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson02_05

クラス名...Item0205Controller

編集するリポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...ItemRepository

ビュー：

フォルダ名...lesson02_05

ファイル名...index.html

利用するテーブル：

テーブル名...item テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_05/index」

実行例：

ID	商品名	価格	ジャンルID
2	みかん	200	1
5	ぶどう	200	1
7	やさしいJava	4000	2
8	やさしいSpring	3000	2
9	やさしいMySQL	4000	2
10	やさしいSeasar2	3000	2
11	やさしいOracle	4000	2

テキスト参考箇所：第 6 章 データベース操作(CRUD) P193 ～ P196

ヒント：

OR 検索を条件とするメソッドは、

`findBy[エンティティのフィールド名 1]Or[エンティティのフィールド名 2]([検索条件の値 1], [検索条件の値 2])`

と定義します。

17 Lesson02_06

ログイン画面で、ユーザーID とパスワード（ともに文字列）を入力するフォームを表示してください。
「ログイン」ボタンが押されたとき、入力されたユーザーID、パスワードと一致するレコードが Lesson02_01 で作成した **training_user テーブル内に存在する場合**、セッション（HttpSession）にユーザーID を記録し、遷移後の画面にそのユーザーID を表示してください。それ以外の場合は、元のログイン画面を再表示してください。また、入力したユーザーID とパスワードは、Form クラスを利用して扱ってください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson02_06

クラス名...Login0206Controller

フォーム：

パッケージ名...jp.co.sss.training.form

クラス名...LoginForm

リポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...TrainingUserRepository

ビュー：

フォルダ名...lesson02_06

ファイル名...login.html、top.html

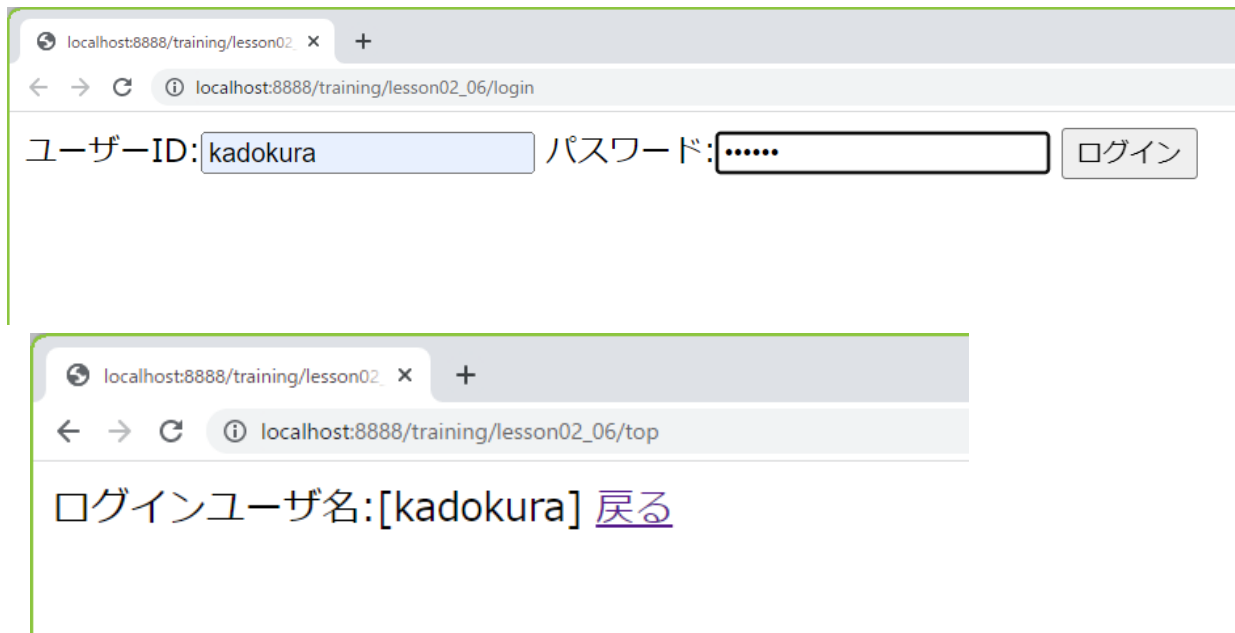
利用するテーブル：

テーブル名...training_user テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_06/login」

実行例

ユーザーIDに「kadokura」、パスワードに「pass01」と入力してログインした場合



The image shows two screenshots of a web browser. The first screenshot shows a login page at `localhost:8888/training/lesson02_06/login`. It has two input fields: 'ユーザーID:' with the value 'kadokura' and 'パスワード:' with masked characters '.....'. A 'ログイン' button is to the right. The second screenshot shows the result page at `localhost:8888/training/lesson02_06/top`. It displays the text 'ログインユーザ名:[kadokura]' followed by a blue link '戻る'.

テキスト参考箇所：第 6 章 データベース操作(CRUD) P193 ～ P196

ヒント：

- ・リポジトリは下記のように定義します。

```
public interface TrainingUserRepository extends JpaRepository<TrainingUser, Long> {  
    TrainingUser findByUserIdAndPassword(String userId, String password);  
}
```

18 Lesson02_07

あいまい検索を行い、データベースに登録されている商品の情報を実行例のような表形式で出力してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson02_07

クラス名...Food0207Controller

エンティティ：

パッケージ名...jp.co.sss.training.entity

クラス名...Food

リポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...FoodRepository

ビュー：

フォルダ名...lesson02_07

ファイル名...index.html

利用するテーブル：

テーブル名...food テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_07/index」

実行する SQL：

- ・下記の SQL を実行し、テーブルを作成してください。

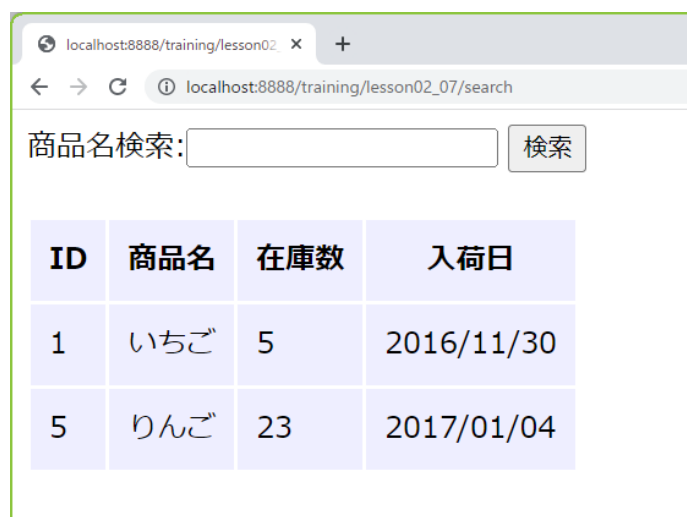
```
CREATE TABLE food (  
  id NUMBER(4) PRIMARY KEY,  
  name VARCHAR2(100) NOT NULL,  
  stock NUMBER(4) NOT NULL,  
  arrival_date DATE DEFAULT SYSDATE  
);  
  
create SEQUENCE seq_food NOCACHE;  
  
INSERT INTO food values (seq_food.NEXTVAL,'いちご',5,'2016-11-30');  
INSERT INTO food values (seq_food.NEXTVAL,'みかん',8,'2016-12-27');  
INSERT INTO food values (seq_food.NEXTVAL,'桃',10,'2016-10-20');  
INSERT INTO food values (seq_food.NEXTVAL,'梨',15,'2016-10-21');  
INSERT INTO food values (seq_food.NEXTVAL,'りんご',23,'2017-01-04');  
  
commit;
```

実行例



ID	商品名	在庫数	入荷日
1	いちご	5	2016/11/30
2	みかん	8	2016/12/27
3	桃	10	2016/10/20
4	梨	15	2016/10/21
5	りんご	23	2017/01/04

入力フォームに「ご」を入力し、「検索」ボタンをクリックした場合



ID	商品名	在庫数	入荷日
1	いちご	5	2016/11/30
5	りんご	23	2017/01/04

テキスト参考箇所：第 6 章 データベース操作(CRUD) P197 ～ P201

ヒント：

・あいまい検索を実施する場合は、「findBy[エンティティのフィールド名]Containing(検索条件の値)」メソッドを使用します。

19 Lesson02_08

Lesson02_01 で作成した training_user テーブルを利用して、下記のようなユーザー登録画面を作成してください。「登録」ボタンが押された場合は、データベースのテーブルに登録処理を行い、詳細画面に遷移して登録したユーザーの情報を表示してください。**※値を入力しなかった場合の考慮はせずに作成して問題ありません。**

登録画面で入力した情報（パラメータ）は、フォームクラスを利用して扱ってください。また、フォームで受け取ったパラメータを「BeanUtils.copyProperties()」メソッドを使用しエンティティにコピーした上で、データベースの登録処理を行ってください。登録処理を行った後は、エンティティから Bean に同メソッドを使用し値をコピーし、画面遷移を行ってください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson02_08

クラス名...User0208Controller

フォーム：

パッケージ名...jp.co.sss.training.form

クラス名...UserForm

ビュー：

フォルダ名...lesson02_08

ファイル名...new.html, show.html

Bean：

パッケージ名...jp.co.sss.training.bean

クラス名...UserBean

エンティティ：

パッケージ名...jp.co.sss.training.enntity

クラス名...TrainingUser

リポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...TrainingUserRepository

利用するテーブル：

テーブル名...training_user テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_08/new」

実行例

ユーザ登録用の画面を表示する。(new.html)

ユーザーを新規登録します。

ユーザーIDとパスワードを入力して、「登録」をクリックしてください。

データベースの「training_user」テーブルで、登録結果を確認してください。

ユーザーID: パスワード:

「登録」ボタンをクリックし、詳細画面を表示する。(show.html)

※「ユーザーID」に「SampleTaro」と入力。8 番目のデータだった場合。

番号	8
ユーザーID	SampleTaro

[戻る](#)

※登録が正常に行われているかどうか、データベースも直接確認をしてください。

テキスト参考箇所：第 6 章 データベース操作(CRUD) P207 ～ P214

ヒント：

フォームクラスは下記の通り作成します。

```
public class UserForm {  
    private String userId;  
    private String password;  
  
    public String getUserId() {  
        return userId;  
    }  
  
    public void setUserId(String userId) {  
        this.userId = userId;  
    }  
  
    public String getPassword() {  
        return password;  
    }  
  
    public void setPassword(String password) {  
        this.password = password;  
    }  
}
```

20 Lesson02_09

Lesson02_01 で作成した training_user テーブルを利用して、下記のようなユーザー情報修正画面を作成してください。「更新（1）」というリンクを押し、「id=1」のユーザー情報を更新する画面を表示してください。その画面でユーザーID とパスワードを入力して「更新」ボタンを押し、データベースのテーブルの「id=1」の行に更新処理を行ってください。更新処理を行った後は、更新を行ったユーザーの詳細情報を表示する画面に遷移してください。※値を入力しなかった場合の考慮はせずに作成して問題ありません。

なお、値の受け渡しについては lesson02_04 と同様 Form クラスと「BeanUtils.copyProperties()」メソッドを使用した流れで作成してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson02_09

クラス名...User0209Controller

フォーム：

パッケージ名...jp.co.sss.training.form

クラス名...UserForm

ビュー：

フォルダ名...lesson02_09

ファイル名...index.html, edit.html, show.html

Bean：

パッケージ名...jp.co.sss.training.bean

クラス名...UserBean

エンティティ：

パッケージ名...jp.co.sss.training.enntity

クラス名...TrainingUser

リポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...TrainingUserRepository

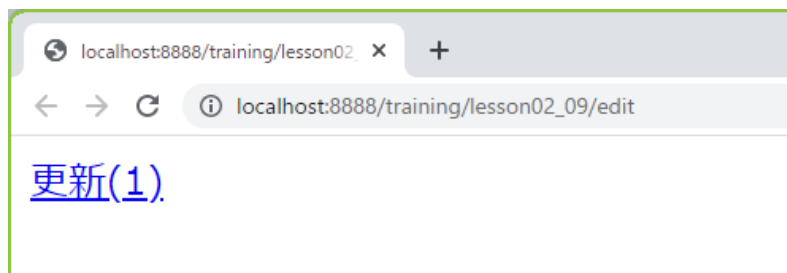
利用するテーブル：

テーブル名...training_user テーブル

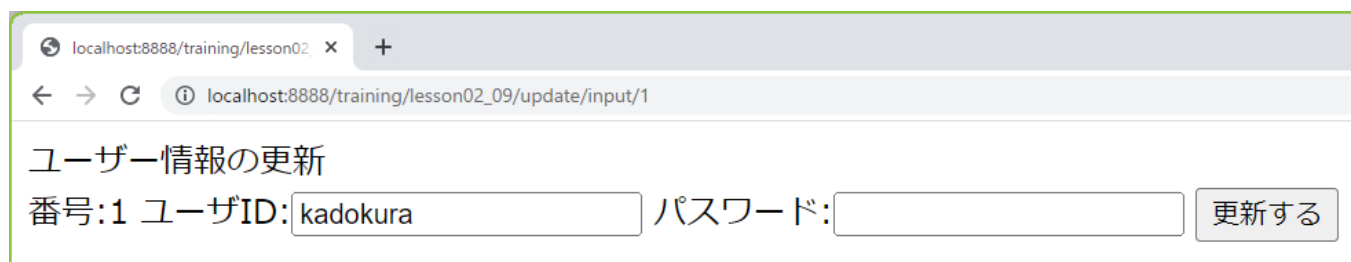
※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_09/edit」

実行例

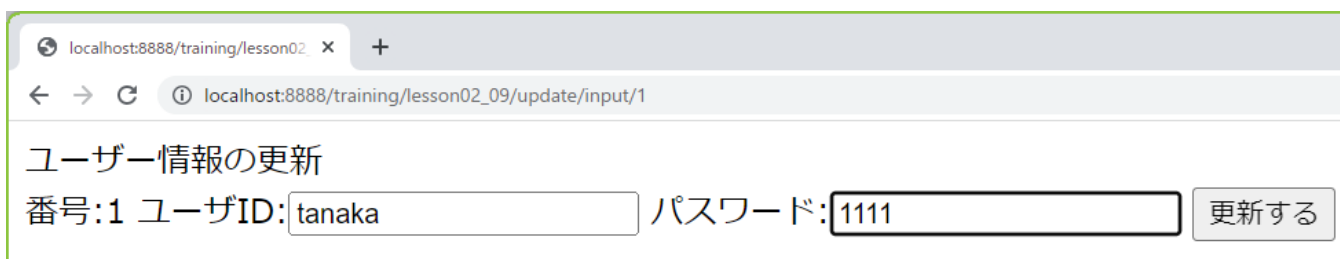
更新用のリンクを表示する画面を表示(index.html)



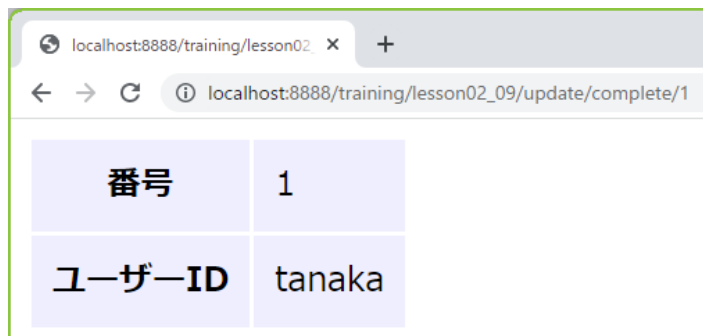
リンクをクリックし、更新用の画面を表示する (edit.html)



ユーザーID「tanaka」パスワード「1111」と入力する。



「更新」をクリックし、詳細画面を表示する (show.html)



番号	1
ユーザーID	tanaka

※更新が正常に行われているかどうか、データベースも直接確認をしてください。

テキスト参考箇所：第6章 データベース操作(CRUD) P215 ～ P222

21 Lesson02_10

Lesson02_07 で作成したプログラムに登録・更新・削除機能を追加したプログラムを作成してください。登録機能は検索画面のリンクから飛ぶことで入力画面に遷移します。また、更新機能は商品名と在庫数の変更を行うことができ、削除機能はボタンを押すことで物理削除を行います。更新・削除機能は表示している商品の右側にボタンを追加してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ :

パッケージ名...jp.co.sss.training.lesson02_10

クラス名...Food0210Controller

フォーム :

パッケージ名...jp.co.sss.training.form

クラス名...FoodForm

ビュー :

フォルダ名...lesson02_10

ファイル名...index.html、insert_input.html、update_input.html、complete.html

Bean :

パッケージ名...jp.co.sss.training.bean

クラス名...FoodeBean

エンティティ :

パッケージ名...jp.co.sss.training.entity

クラス名...Food

リポジトリ :

パッケージ名...jp.co.sss.training.repository

インターフェイス名...FoodRepository

利用するテーブル :

テーブル名...food テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_10/index」

実行例

商品名検索: [新規登録](#)

ID	商品名	在庫数	入荷日	更新	削除
1	いちご	5	2016/11/30	<input type="button" value="更新"/>	<input type="button" value="削除"/>
2	みかん	8	2016/12/27	<input type="button" value="更新"/>	<input type="button" value="削除"/>
3	桃	10	2016/10/20	<input type="button" value="更新"/>	<input type="button" value="削除"/>
4	梨	15	2016/10/21	<input type="button" value="更新"/>	<input type="button" value="削除"/>
5	りんご	23	2017/01/04	<input type="button" value="更新"/>	<input type="button" value="削除"/>

↓ 「新規登録」をクリック

商品名: 在庫数:

↓ 商品名と在庫数を入力して、登録ボタンをクリック

処理が完了しました

[戻る](#)

↓ 「戻る」をクリック

商品名検索: [新規登録](#)

ID	商品名	在庫数	入荷日	更新	削除
1	いちご	5	2016/11/30	<input type="button" value="更新"/>	<input type="button" value="削除"/>
2	みかん	8	2016/12/27	<input type="button" value="更新"/>	<input type="button" value="削除"/>
3	桃	10	2016/10/20	<input type="button" value="更新"/>	<input type="button" value="削除"/>
4	梨	15	2016/10/21	<input type="button" value="更新"/>	<input type="button" value="削除"/>
5	りんご	23	2017/01/04	<input type="button" value="更新"/>	<input type="button" value="削除"/>
6	キャベツ	4	2017/01/04	<input type="button" value="更新"/>	<input type="button" value="削除"/>

更新ボタンをクリック（サンプルでは ID:6 のジャガイモを選択）

商品名: 在庫数:

↓ 商品名と在庫数を入力して、登録ボタンをクリック

処理が完了しました

[戻る](#)

↓ 「戻る」をクリック

商品名検索:

検索

[新規登録](#)

ID	商品名	在庫数	入荷日	更新	削除
1	いちご	5	2016/11/30	<input type="button" value="更新"/>	<input type="button" value="削除"/>
2	みかん	8	2016/12/27	<input type="button" value="更新"/>	<input type="button" value="削除"/>
3	桃	10	2016/10/20	<input type="button" value="更新"/>	<input type="button" value="削除"/>
4	梨	15	2016/10/21	<input type="button" value="更新"/>	<input type="button" value="削除"/>
5	りんご	23	2017/01/04	<input type="button" value="更新"/>	<input type="button" value="削除"/>
6	じゃがいも	17	2017/01/04	<input type="button" value="更新"/>	<input type="button" value="削除"/>

削除ボタンをクリック（サンプルでは ID:2 のみかんを選択）

処理が完了しました

[戻る](#)

↓ 「戻る」をクリック

商品名検索:

検索

[新規登録](#)

ID	商品名	在庫数	入荷日	更新	削除
1	いちご	5	2016/11/30	更新	削除
3	桃	10	2016/10/20	更新	削除
4	梨	15	2016/10/21	更新	削除
5	りんご	23	2017/01/04	更新	削除
6	じゃがいも	17	2017/01/04	更新	削除

テキスト参考箇所：第 6 章 データベース操作(CRUD) P207 ～ P214, P215 ～ P222, P223～P228

22 Lesson02_11

Lesson02_02 で作成した item テーブルを利用して、価格の高い順に並び替え、その先頭 5 件について、結果を表形式で出力してください。検索結果の先頭 5 件を降順で取得するメソッドは、「**findTop5ByOrderBy[エンティティのフィールド名]Desc()**」と定義します。

記述にあたっては以下の仕様に従うものとします。

コントローラ :

パッケージ名...jp.co.sss.training.lesson02_11

クラス名...Item0211Controller

編集するリポジトリ :

パッケージ名...jp.co.sss.training.repository

インターフェイス名...ItemRepository

ビュー :

フォルダ名...lesson02_11

ファイル名...index.html

利用するテーブル :

テーブル名...item テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_11/index」

実行例 :

ID	商品名	価格	ジャンルID
7	やさしいJava	4000	2
9	やさしいMySQL	4000	2
11	やさしいOracle	4000	2
8	やさしいSpring	3000	2
10	やさしいSeasar2	3000	2

テキスト参考箇所 : 第 6 章 データベース操作(CRUD) P188 ~ P201

23 Lesson02_12

Lesson02_02 で作成した item テーブルを利用して、商品名が「やさしい」で始まる商品を検索し、結果を表形式で出力してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson02_12

クラス名...Item0212Controller

編集するリポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...ItemRepository

ビュー：

フォルダ名...lesson02_12

ファイル名...index.html

利用するテーブル：

テーブル名...item テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson02_12/index」

実行例：

ID	商品名	価格	ジャンルID
7	やさしいJava	4000	2
8	やさしいSpring	3000	2
9	やさしいMySQL	4000	2
10	やさしいSeasar2	3000	2
11	やさしいOracle	4000	2

6

テキスト参考箇所：第 6 章 データベース操作(CRUD) P197 ～ P201

ヒント:

前方一致で検索するメソッドは、

`findBy[エンティティのフィールド名]StartingWith(検索条件の値)` と定義します。

24 Lesson03_01

社員（社員番号、社員 ID）の情報と、社員に紐づく部署の情報（部署 ID、部署名）を一覧で表示してください。

データベースには、新しく外部参照元となる「社員」テーブル（employee）と、外部参照先となる「部署」テーブル（department）を作成してください。また、エンティティを用いて「社員：部署」で「多対 1」の外部参照関係の関連付けにすること。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson03_01

クラス名...Employee0301Controller

エンティティ：

パッケージ名...jp.co.sss.training.lesson03_01

クラス名...employee

リポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...EmployeeRepository

ビュー：

フォルダ名...lesson03_01

ファイル名...index.html

利用するテーブル：

テーブル名...employee テーブル、 department テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson03_01/index」

使用する SQL :

下の SQL 文で「department」及び「employee」テーブルを作成し、レコードを登録する。

※「department（部署テーブル）」→「employee（社員テーブル）」の順番で作成してください。

```
-- 部署テーブル作成
CREATE TABLE department (
  id NUMBER(3) PRIMARY KEY,
  name VARCHAR2(100)
);

-- 部署データの登録
CREATE SEQUENCE seq_department NOCACHE;
INSERT INTO department VALUES (seq_department.NEXTVAL, '開発部');
INSERT INTO department VALUES (seq_department.NEXTVAL, '営業部');

-- 社員テーブル作成
CREATE TABLE employee(
  id NUMBER(4) PRIMARY KEY,
  user_id VARCHAR2(10),
  department_id NUMBER(3) NOT NULL REFERENCES department(id)
);

-- 社員テーブルデータの登録
CREATE SEQUENCE seq_employee NOCACHE;
INSERT INTO employee VALUES (seq_employee.NEXTVAL,'user01',1);
INSERT INTO employee VALUES (seq_employee.NEXTVAL,'user02',2);
INSERT INTO employee VALUES (seq_employee.NEXTVAL,'user03',2);
INSERT INTO employee VALUES (seq_employee.NEXTVAL,'user04',1);
INSERT INTO employee VALUES (seq_employee.NEXTVAL,'user05',2);
INSERT INTO employee VALUES (seq_employee.NEXTVAL,'user06',1);

commit;
```

実行例 :

社員番号	社員ID	部署ID	部署名
1	user01	1	開発部
2	user02	2	営業部
3	user03	2	営業部
4	user04	1	開発部
5	user05	2	営業部
6	user06	1	開発部

テキスト参考箇所：第 7 章 データベース操作(外部参照) P234～P247

ヒント：

- ・「department」テーブルが「外部参照先」となるため、外部参照元テーブル「employee」のエンティティに該当する「Employee」エンティティには、外部参照先テーブルのエンティティ「Department」をフィールドとして定義します。

25 Lesson03_02

Lesson03_01 で作成した「employee」テーブルの外部キー「department_id」による条件検索を行い、部署 ID に紐づく社員の情報を一覧で表示してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson03_02

クラス名...User0302Controller

エンティティ：

パッケージ名...jp.co.sss.training.entity

クラス名...Employee、Department

リポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...EmployeeRepository

ビュー：

フォルダ名...lesson03_02

ファイル名...index.html

利用するテーブル：

テーブル名...lesson04_05user テーブル、department テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson03_02/index/1」

実行例

※入力した URL の末尾 (…/index/1) の「1」の部分を変更した場合、「部署 ID」が「2」の社員情報が表示されることも確認しましょう。

社員番号	社員ID	部署ID	部署名
1	user01	1	開発部
4	user04	1	開発部
6	user06	1	開発部

テキスト参考箇所：第 7 章 データベース操作(外部参照) P248～P251

26 Lesson04_01

データベースから取得したユーザーデータを表形式で表示してください。テキストボックスに文字列を入力して「検索」を押した場合に、入力された文字列を含むユーザーを検索するあいまい検索を実行し、その結果を表示してください。

検索は「@NamedQuery」アノテーションを使用して実装してください。（Lesson02_01 で作成した training_user テーブルを利用してください）

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson04_01

クラス名...User0401Controller

編集するエンティティ：

パッケージ名...jp.co.sss.training.entity

クラス名...TrainingUser

ビュー：

フォルダ名...lesson04_01

ファイル名...index.html

利用するテーブル：

テーブル名...training_user テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson04_01/index」

実行例：（初期表示）

localhost:8888/training/lesson04_01/index

ユーザー名（あいまい検索）： 検索

ID	ユーザー名
1	tanaka
2	yamamoto
3	ogawa
4	kuramae
5	yamashita
6	kanagawa
7	SampleTaro

テキストボックスに「kura」と入力して、検索ボタンをクリック

検索後の URL：「http://localhost:8888/training/lesson04_01/searchName?keyword=kura」

ユーザー名（あいまい検索）： 検索

ID	ユーザー名
4	kuramae

テキスト参考箇所：第 8 章 データベース操作(JPQL) P263 ～ P268

ヒント：

- ・ @NamedQuery アノテーションは「エンティティ」の中に定義します。
 - ・ あいまい検索を JPQL で作成する場合は「LIKE」キーワードを使用します。
- ※例えば、とあるキーワードに基づいてあいまい検索を実行する場合、JPQL は
- 「SELECT 別名 FROM エンティティ名 別名 WHERE 別名.フィールド名 LIKE [パラメータ]」
- という形になります。
- ・ JPQL でもあいまい検索を実行する場合、「0 文字以上の任意の文字列」を示す「%」が使えます。

27 Lesson04_02

データベースから取得したユーザーデータを表形式で表示してください。テキストボックスに文字列を入力して「検索」を押した場合に、入力された文字列を含むユーザーを検索するあいまい検索を実行し、その結果を表示してください。

検索は「@Query」アノテーションを使用して実装してください。（Lesson02_01 で作成した training_user テーブルを利用してください）

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson04_02

クラス名...User0402Controller

編集するリポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...TrainingUserRepository

ビュー：

フォルダ名...lesson04_01

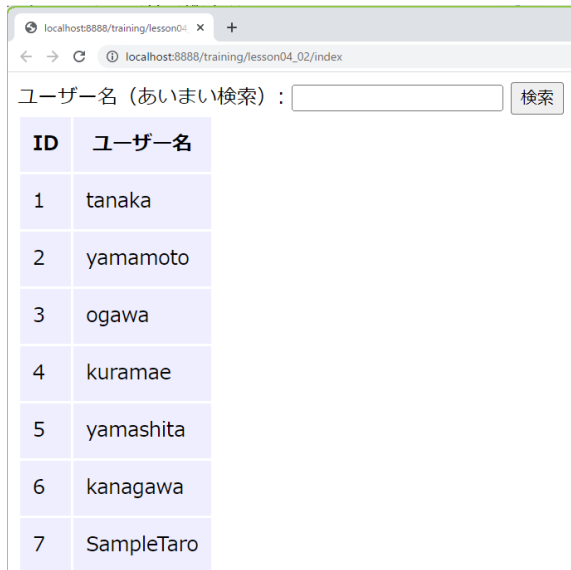
ファイル名...index.html

利用するテーブル：

テーブル名...training_user テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson04_02/index」

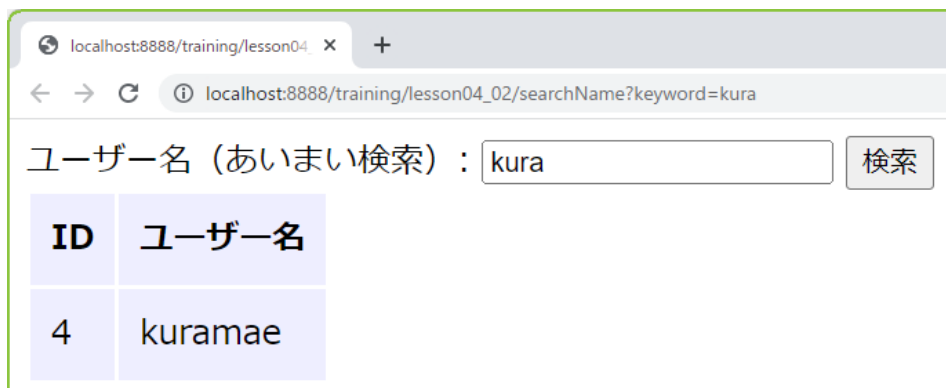
実行例



ID	ユーザー名
1	tanaka
2	yamamoto
3	ogawa
4	kuramae
5	yamashita
6	kanagawa
7	SampleTaro

テキストボックスに「kura」と入力して、検索ボタンをクリック

検索後の URL : 「http://localhost:8888/training/lesson04_02/searchName?keyword=kura」



ID	ユーザー名
4	kuramae

テキスト参考箇所 : 第 8 章 データベース操作(JPQL) P269 ~ P273

ヒント:

- @Query アノテーションは「リポジトリ」の中に定義します。
- @NamedQuery で実装した場合と、組み立てる JPQL は同一になります。

28 Lesson04_03

テキストボックスにユーザー名の一部を入力して「検索」を押した場合に、入力された文字列を含むユーザーを検索してください。その際、検索結果に該当する件数を「該当ユーザー数：●人」という形式で表示をした上で、ユーザー一覧の検索結果を表示してください。

検索は「@Query」アノテーションを使用して実装してください。（Lesson02_01 で作成した training_user テーブルを利用してください）

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson04_03

クラス名...User0403Controller

編集するリポジトリ：

パッケージ名...jp.co.sss.training.repository

インターフェイス名...TrainingUserRepository

ビュー：

フォルダ名...lesson04_03

ファイル名...index.html

利用するテーブル：

テーブル名...training_user テーブル

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson04_03/index」

実行例：（初期表示：ユーザーが 8 名の場合）

ユーザー名（あいまい検索）：

ID	ユーザー名
1	tanaka
2	yamamoto
3	ogawa
4	kuramae
5	yamashita
6	kanagawa
7	SampleTaro

「y」と入力し、「yamamoto」と「yamashita」の 2 件が該当した場合

ユーザー名（あいまい検索）：

該当ユーザー数: 2人

ID	ユーザー名
2	yamamoto
5	yamashita

テキスト参考箇所：第 8 章 データベース操作(JPQL) P259, P269 ～ P273

ヒント：

- ・件数を取得する検索を JPQL で作成する場合は「COUNT」キーワードを使用します。

※例えば、とあるキーワードに基づいて検索結果の件数を取得する場合、JPQL は

「SELECT COUNT(別名) FROM エンティティ名 別名 WHERE 別名.フィールド名 LIKE [パラメータ] 」
という形になります。

- ・検索結果の件数を取得するメソッドの戻り値の型は、数値(int)を使用します。

29 Lesson05_01

ユーザーID とパスワードを入力するフォームを実装してください。ユーザーID は小文字アルファベットか数字（先頭に数字は使えない）のみ受け付けます。また、ユーザーID ・パスワードともに 1 文字以上 10 文字以下の入力のみ受け付けます。チェックルールは@Size や@Pattern アノテーションで実装してください。入力された情報はデータベースに登録する必要はありません。

この問題では、下記のとおり Form 内に定義するアノテーションの属性「message」を使用してメッセージを定義してください。

```
public class UserForm {

    @Size(min = 1, max = 10, message = "1文字以上10文字以下で入力してください")
    @Pattern(regexp = "[a-z][a-z0-9]*", message = "小文字アルファベットまたは数字を入力してください。先頭には数字は使えません")
    private String userId;
```

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson05_01

クラス名...User0501Controller

フォーム：

パッケージ名...jp.co.sss.training.lesson05_01

クラス名...User0501Form

ビュー：

フォルダ名...lesson05_01

ファイル名...new.html、result.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson05_01/new」

実行例

ユーザーID:

パスワード:

登録ボタンをクリック（入力チェックを行い、エラーメッセージを表示）

ユーザーID: 小文字アルファベットまたは数字を入力してください。先頭には数字は使えません
1文字以上10文字以下で入力してください

パスワード: 1文字以上10文字以下で入力してください

適切なユーザーID、パスワードを入力して、登録ボタンをクリック

ユーザーの登録が完了しました。 [別のユーザーを登録](#)

テキスト参考箇所：第 9 章 入力チェック P278～P296

30 Lesson05_02

ユーザーID とパスワードを入力するフォームを実装してください。ユーザーID は小文字アルファベットか数字（先頭に数字は使えない）のみ受け付けます。また、ユーザーID ・パスワードともに 1 文字以上 10 文字以下の入力のみ受け付けます。チェックルールは@Size や@Pattern アノテーションで実装してください。入力された情報はデータベースに登録する必要はありません。

エラーメッセージは、設定ファイル（ValidationMessages.properties）に記述してください。入力された情報はデータベースに登録する必要はありません。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson05_02

クラス名...User0502Controller

フォーム：

パッケージ名...jp.co.sss.training.lesson05_02

クラス名...User0502Form

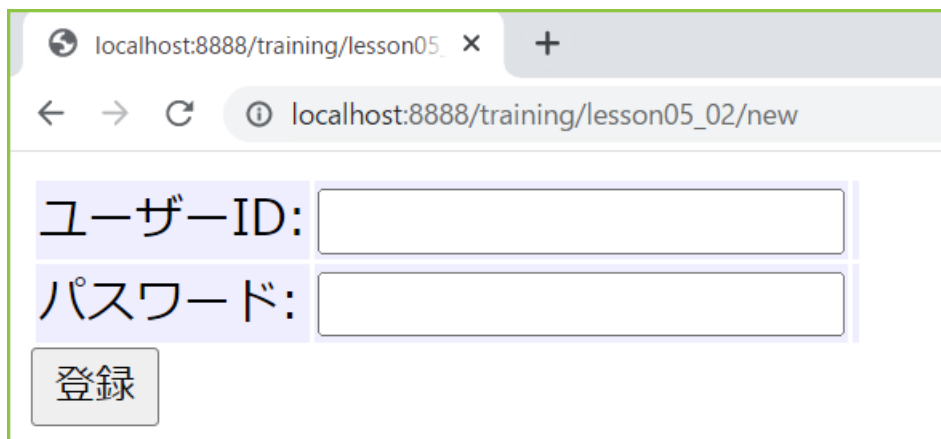
ビュー：

フォルダ名...lesson05_02

ファイル名...new.html、result.html

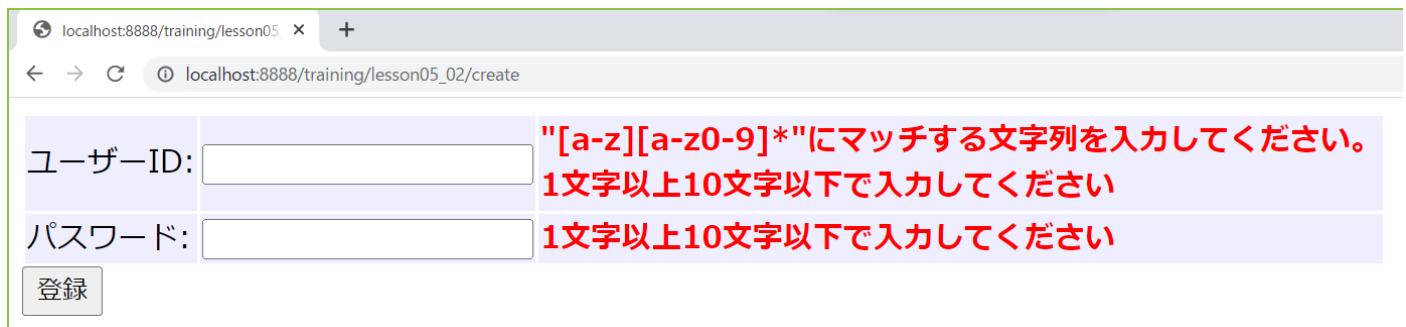
※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson05_02/new」

実行例



The screenshot shows a web browser window with the address bar displaying 'localhost:8888/training/lesson05_02/new'. The page content includes a form with two input fields. The first field is labeled 'ユーザーID:' and the second is labeled 'パスワード:'. Below these fields is a button labeled '登録' (Register).

すべて空白の状態で「登録」ボタンをクリック



The screenshot shows a web browser window with the address bar displaying "localhost:8888/training/lesson05_02/create". The page contains a registration form with two input fields: "ユーザーID:" and "パスワード:". To the right of these fields, there are red validation messages. Below the fields is a "登録" (Register) button.

ユーザーID:	<input type="text"/>	"[a-z][a-z0-9]*"にマッチする文字列を入力してください。 1文字以上10文字以下で入力してください
パスワード:	<input type="password"/>	1文字以上10文字以下で入力してください

登録

テキスト参考箇所：

第9章 入力チェック P278～P296

第10章 メッセージ出力 P298～P308

ヒント：

設定ファイル「ValidationMessages.properties」の実装例

```
javax.validation.constraints.Size.message={min}文字以上{max}文字以下で入力してください  
javax.validation.constraints.Pattern.message="{regex}"にマッチする文字列を入力してください。
```


31 Lesson06_01

フィルターを使用し、リクエスト（コントローラのメソッド）の処理前に、「フィルターが実行されました」というメッセージをコンソールに出力してください。

記述にあたっては以下の仕様に従うものとします。

フィルター：

パッケージ名...jp.co.sss.training.lesson06_01

クラス名...Message0601Filter

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_01/hello」

こんにちは

↓コンソール出力

```

問題 @ Javadoc 宣言 コンソール 進行状況
spring_training_answer - SpringTrainingAnswerApplication [Spring Boot アプリケーション] C:
2020-12-02 10:23:57.348 INFO 9764 --- [ restartedMain] o
2020-12-02 10:23:57.368 INFO 9764 --- [ restartedMain] o
2020-12-02 10:23:57.368 INFO 9764 --- [ restartedMain] o
2020-12-02 10:23:57.368 INFO 9764 --- [ restartedMain] o
2020-12-02 10:23:57.368 INFO 9764 --- [ restartedMain] o
2020-12-02 10:23:57.626 INFO 9764 --- [ restartedMain] o
2020-12-02 10:23:57.628 INFO 9764 --- [ restartedMain] j
2020-12-02 10:23:58.314 INFO 9764 --- [ restartedMain] o
2020-12-02 10:23:58.325 WARN 9764 --- [ restartedMain] a
2020-12-02 10:23:58.558 INFO 9764 --- [ restartedMain] o
2020-12-02 10:23:58.587 INFO 9764 --- [ restartedMain] o
2020-12-02 10:23:58.588 INFO 9764 --- [ restartedMain] j
2020-12-02 10:23:58.590 INFO 9764 --- [ restartedMain] .
2020-12-02 10:24:09.586 INFO 9764 --- [nio-8888-exec-1] o
2020-12-02 10:24:09.587 INFO 9764 --- [nio-8888-exec-1] o
2020-12-02 10:24:09.598 INFO 9764 --- [nio-8888-exec-1] o
フィルターが実行されました

```

テキスト参考箇所：第 11 章 フィルタ P329 ～P333

32 Lesson06_02

「2 番目のフィルターが実行されました」とコンソールに出力するフィルターを作成してください。
その上で、Lesson06_01 で作成したフィルター→今回作成したフィルタの順で必ず実行されるようにしてください。

記述にあたっては以下の仕様に従うものとします。

フィルター：

パッケージ名...jp.co.sss.training.lesson06_02

クラス名...Message0602Filter

編集するフィルター：

パッケージ名...jp.co.sss.training.lesson06_01

クラス名...Message0601Filter

設定クラス：

パッケージ名...jp.co.sss.training.lesson06_02

クラス名...Filter0602Config

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson01_01/hello」

こんにちは

↓コンソール出力

```

spring_training_answer - SpringTrainingAnswerApplication [Spring Boot アプリケーション]
2020-12-02 10:32:56.879 INFO 9764 --- restartedMai
2020-12-02 10:32:56.902 INFO 9764 --- restartedMai
2020-12-02 10:32:56.902 INFO 9764 --- restartedMai
2020-12-02 10:32:56.904 INFO 9764 --- restartedMai
2020-12-02 10:33:14.602 INFO 9764 --- nio-8888-exec
2020-12-02 10:33:14.602 INFO 9764 --- nio-8888-exec
2020-12-02 10:33:14.614 INFO 9764 --- nio-8888-exec
フィルターが実行されました
2番目のフィルターが実行されました
  
```

テキスト参考箇所：第 11 章 フィルタ P334 ～P341

33 Lesson07_01

Thymeleaf の「th:attr」と「th:text」を使用して、下記のような画面を出力してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ :

パッケージ名...jp.co.sss.training.lesson07_01

クラス名...Index0701Controller

ビュー :

フォルダ名...lesson07_01

ファイル名...index.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson07_01/index」

実行例 :

処理が完了しました

テキスト参考箇所 : 第 12 章 Thymeleaf P352 ~P353

ヒント:

出力される HTML は次のようになります（注意：下記をそのまま HTML ファイルに書くわけではなく、「information」や「処理が完了しました」という部分を Thymeleaf の機能を用いて埋め込むようにしてください）

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title></title>
    <style>
      .information { color: red; }
    </style>
  </head>
  <body>
    <div class="information">処理が完了しました</div>
  </body>
</html>
```

例えば、上記ソースコードの<div class="information">処理が完了しました。</div>の箇所の「class」属性については、「th:attr」を使用して「<div th:attr="class=\${スコープ変数名}"...>」のように属性を追加します。

34 Lesson07_02

Thymeleaf の「th:if」を使用して、入力した検索キーワードの内容に応じてメッセージを出力してください。検索キーワードが入力されていない場合は、背景色を赤くした状態で「検索キーワードを入力してください」、1文字以上入力した場合は背景色を緑にした状態で「入力された検索キーワードは[入力した検索キーワード]です」と表示してください。背景色は厳密に指定しない、また定義する記述は HTML の<style>タグで良いこととする。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson07_02

クラス名...Index0702Controller

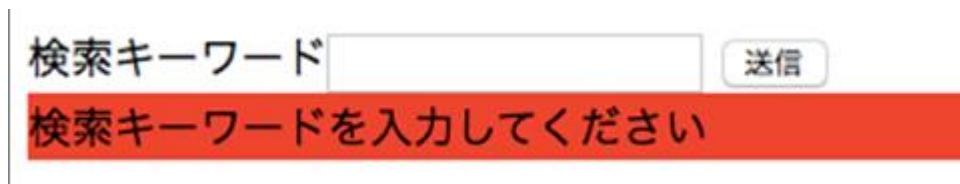
ビュー：

フォルダ名...lesson07_02

ファイル名...index.html

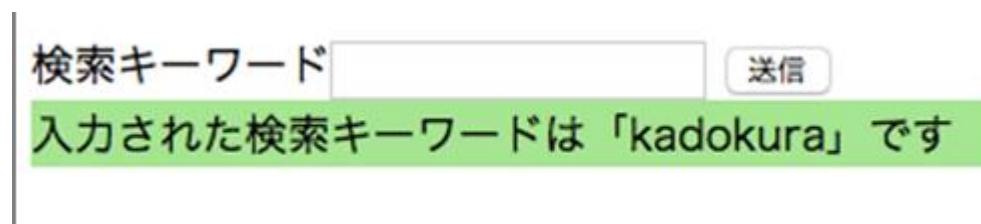
※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson07_02/index」

実行例：



A screenshot of a web form. At the top, there is a label '検索キーワード' (Search Keyword) followed by a text input field. To the right of the input field is a button labeled '送信' (Send). Below the input field, a red banner displays the message '検索キーワードを入力してください' (Please enter a search keyword).

↓キーワードに「kadokura」を入力して「送信」をクリックした場合



A screenshot of the same web form after a successful search. The input field now contains the text 'kadokura'. The '送信' button is still present. Below the input field, a green banner displays the message '入力された検索キーワードは「kadokura」です' (The entered search keyword is 'kadokura').

テキスト参考箇所：第 12 章 Thymeleaf P360

35 Lesson07_03

Thymeleaf の「th:text」を使用して、セッション（HttpSession）に格納されたカウンタの値を表示してください。カウンタの値はアクセスするたびに 1 ずつ増やしてください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson07_03

クラス名...Index0703Controller

ビュー：

フォルダ名...lesson07_03

ファイル名...index.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson07_03/index」

実行例：（このページをリロードする度に「3」の部分の数値が増えていく）

3

テキスト参考箇所：第 12 章 Thymeleaf P363

ヒント：

「th:text」からセッションにアクセスするには、「#httpSession」を使用します。

```
<div th:text="${#httpSession.getAttribute('count')}"></div>
```

36 Lesson07_04

css を利用して、テキストボックスの背景色を黄色に設定してください。また、氏名が未入力状態で「送信」ボタンをクリックしたときは、JavaScript を利用して、「氏名が入力されていません」というダイアログを表示してください。CSS は「style.css」、JavaScript は「common.js」ファイルに記述し、Thymeleaf の「th:href」、「th:src」を用いてリンクしてください。

記述にあたっては以下の仕様に従うものとします。

コントローラ :

パッケージ名...jp.co.sss.training.lesson07_04

クラス名...Index0704Controller

ビュー :

フォルダ名...lesson07_04

ファイル名...index.html

CSS :

フォルダ名...css

ファイル名...style.css

JavaScript :

フォルダ名 : js

ファイル名 : common.js

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson07_04/index」

実行例

氏名を入力して「送信」ボタンを押してください。

氏名:

氏名に何も入力せずに「送信」ボタンをクリックした場合



テキスト参考箇所：第 12 章 Thymeleaf P367～373

ヒント：

```
<link th:href="@{/css/style.css}" rel="stylesheet" />
<script th:src="@{/js/common.js}"></script>
```

Thymeleaf から CSS、JavaScript ファイルへリンクするには下記のような記述を利用します。

また、CSS ファイルは「/src/main/resources/static/css」以下、JavaScript ファイルは「/src/main/resources/static/js」以下に配置します。

37 Lesson07_05

※講義では触れていない応用問題です。

Thymeleaf のレイアウト機能を用いて、「社員一覧」画面と「社員新規登録」画面のヘッダー部分（タイトル「社員管理システム」と、ナビゲーションリンク、水平線）、フッター部分（水平線、著作権表示）を出力するようにしてください。（ページのみ実装。実際の一覧処理や登録処理は動作しなくてよい）
（画面は index.html、layout.html、new.html の 3 つのファイルを使用して実装してください）

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson07_05

クラス名...Index0705Controller

ビュー：

フォルダ名...lesson07_05

ファイル名...index.html、layout.html、header.html、footer.html、new.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson07_05/index」

実行例：

社員管理システム	
社員一覧 新規社員登録	
社員番号	社員名
1	yamamoto
2	kadokura
Copyright © 2017 System Shared. All rights reserved.	

「新規社員登録」リンクをクリック時

社員管理システム	
社員一覧 新規社員登録	
社員名:	<input type="text"/> <input type="button" value="登録"/>
Copyright © 2017 System Shared. All rights reserved.	

テキスト参考箇所：第 12 章 Thymeleaf P383～391

ヒント:

ヘッダーとフッターの出力部分はレイアウトファイルに記述し、「社員一覧」画面と「社員新規登録」画面に固有の出力はそれぞれ対応するファイルから出力します。

```
<body>
  <header>
    <h1>社員管理システム</h1>
    <a href="index">社員一覧</a>
    <a href="new">新規社員登録</a>
    <hr/>
  </header>
  <div layout:fragment="main" th:remove="tag"></div>
  <footer>
    <hr/>
    Copyright &copy; 2017 System Shared. All rights reserved.
  </footer>
</body>
```

38 Lesson08_01

※講義では触れていない応用問題です。

Lesson02_01 の一覧画面にページングを実装してください。1 ページあたりの表示件数は 5 件としてください。

(Lesson02_01 で作成した training_user テーブルを利用して、あらかじめレコードを 6 件以上登録してから実行してください)

記述にあたっては以下の仕様に従うものとします。

コントローラ :

パッケージ名...jp.co.sss.training.lesson08_01

クラス名...User0801Controller

設定クラス

パッケージ名...jp.co.sss.training.lesson08_01

クラス名...lesson08_01Config

ビュー :

フォルダ名...lesson08_01

ファイル名...index.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson08_01/indexPaging」

実行例 : 1 ページ目


番号	ユーザーID
1	tanaka
2	yamamoto
3	ogawa
4	kuramae
5	yamashita

<< 1 2 >>

実行例：2 ページ目

番号	ユーザーID
6	kanagawa

<< 1 2 >>



テキスト参考箇所：第 13 章 ページング P397～419

39 Lesson08_02

※講義では触れていない応用問題です。

Lesson01_01 と同様のメッセージを表示してください。この処理の過程で、インターセプターを使用し、リクエスト（コントローラのメソッド）の処理前に、「リクエストを処理します」というメッセージをコンソールに出力してください。

記述にあたっては以下の仕様に従うものとします。

コントローラ：

パッケージ名...jp.co.sss.training.lesson08_02

クラス名...Hello0802Controller

アスペクト：

パッケージ名...jp.co.sss.training.lesson08_02

クラス名...HelloInterceptor

ビュー：

フォルダ名...lesson08_02

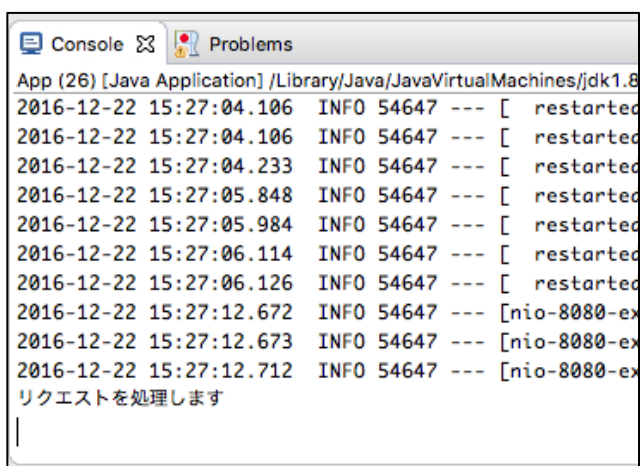
ファイル名...hello.html

※Web ブラウザの URL 欄に入力する URL 「http://localhost:8888/training/lesson08_02/hello」

実行例

こんにちは

コンソール出力



```

App (26) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8
2016-12-22 15:27:04.106 INFO 54647 --- [ restarted
2016-12-22 15:27:04.106 INFO 54647 --- [ restarted
2016-12-22 15:27:04.233 INFO 54647 --- [ restarted
2016-12-22 15:27:05.848 INFO 54647 --- [ restarted
2016-12-22 15:27:05.984 INFO 54647 --- [ restarted
2016-12-22 15:27:06.114 INFO 54647 --- [ restarted
2016-12-22 15:27:06.126 INFO 54647 --- [ restarted
2016-12-22 15:27:12.672 INFO 54647 --- [nio-8080-ex
2016-12-22 15:27:12.673 INFO 54647 --- [nio-8080-ex
2016-12-22 15:27:12.712 INFO 54647 --- [nio-8080-ex
リクエストを処理します

```

テキスト参考箇所：第 15 章 AOP P440 ～P445

ヒント：

インターセプターのクラスは下記のように実装します。

```
@Component↓
@Aspect↓
public class HelloInterceptor {↓
    ↓
    @Before("execution(* jp.co.sss.practice.lesson06_01.*.*(..))")↓
    public void before(JoinPoint jp) {↓
        System.out.println("リクエストを処理します");↓
    }↓
}↓
```