フレームワーク基礎実習

05. Thymeleaf 入門

株式会社ジードライブ

今回学ぶこと

- Thymeleafの導入
- コントローラーとThymeleafの連携
- 参考：静的サイトからの作業手順

Thymeleafとは

- HTMLやXMLを生成することができるテンプレートエンジン
  - 通常のHTMLに加え、 Thymeleaf特有のタグや属性を使うことができ、コントローラーから渡されたデータを処理することができる

例：コントローラーで生成した商品情報をHTMLのリスト形式で出力する

コントローラー

```
@GetMapping("/items")
public String showItemList() {
   List<String> itemList = Arrays.asList("葡萄", "桃", "柿");
   model.addAttribute("itemList", itemList);
   return "itemListPage";
}
```

Thymeleaf

```
<h2>商品一覧</h1>
<ul>
  <li th:each="item : ${itemList}" th:text="${item}"></li>
</ul>
```

Thymeleafの導入

① プロジェクト作成時に、
依存関係としてThymeleaf
を加える

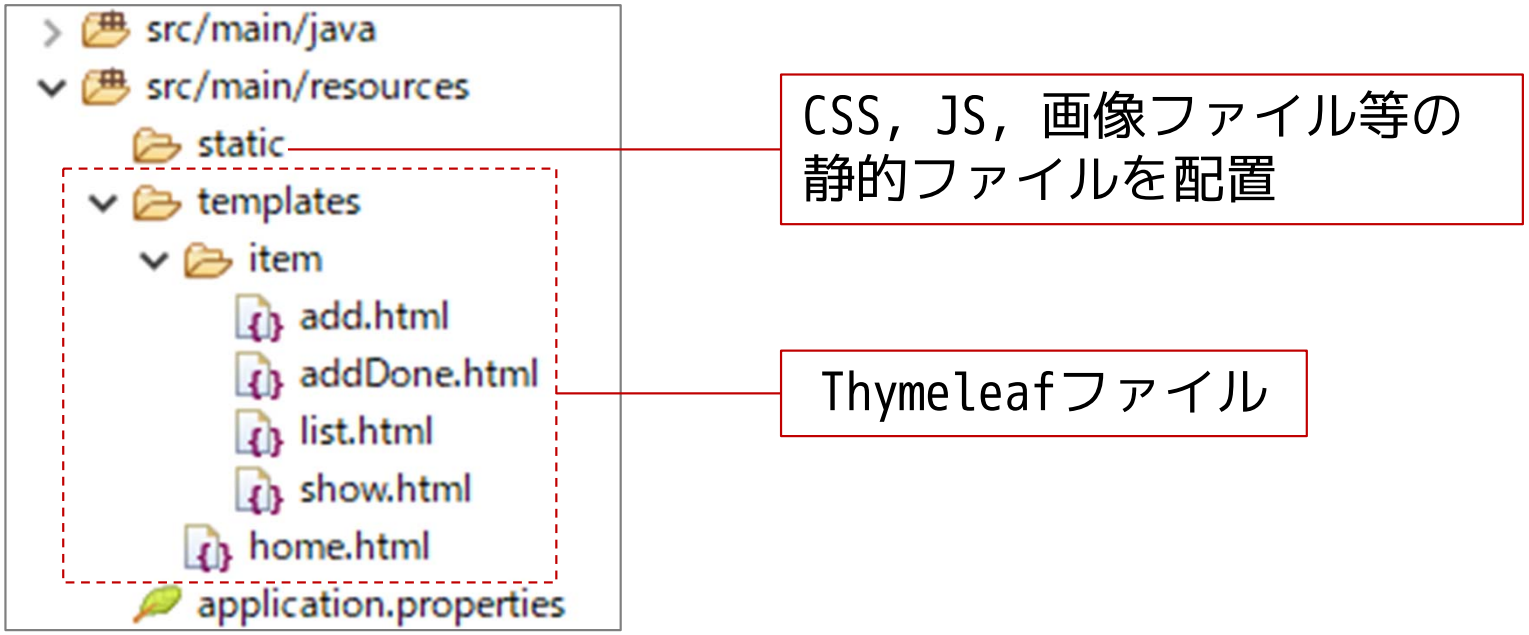
② Thymeleafファイルに名前空間を追記する

```
<!doctype html>
<html lang="ja" xmlns:th="http://www.thymeleaf.org">
<head>
…
```

th:〇〇 といった形式でThymeleafタグや属性を
利用することができる（thにすることが一般的）

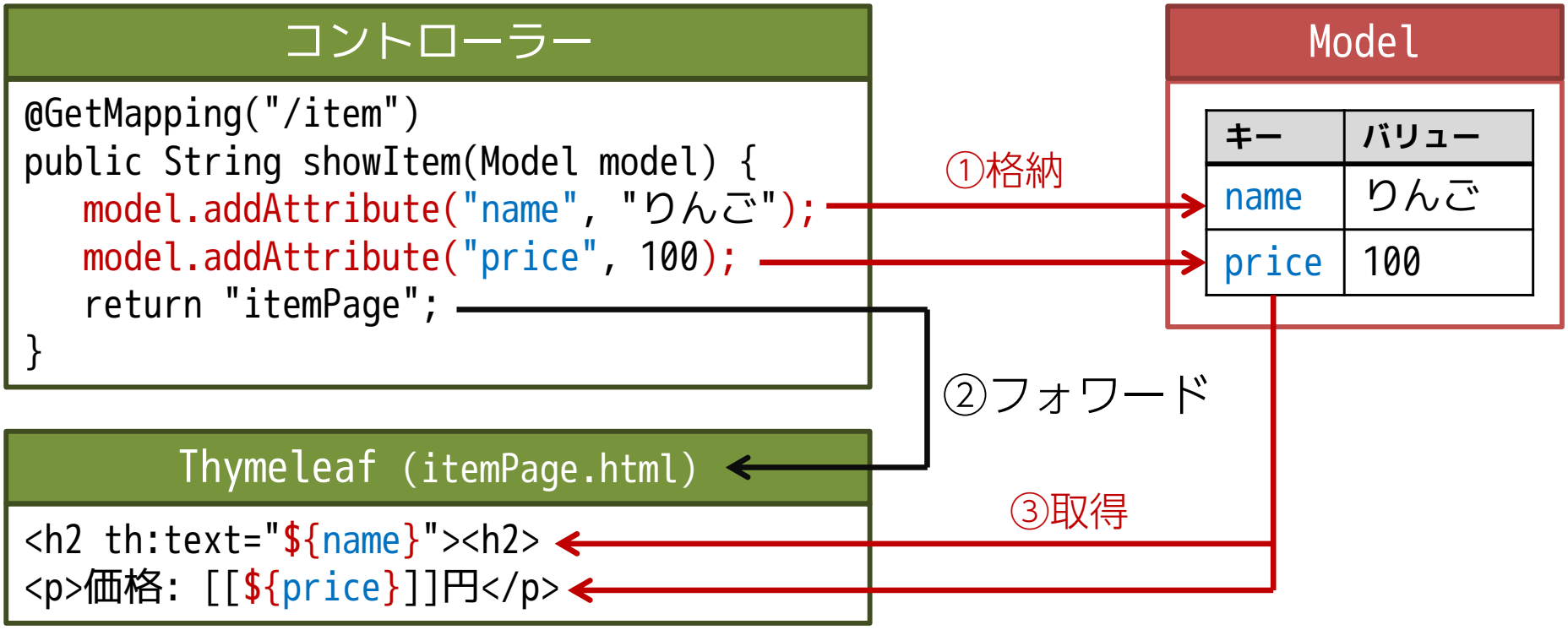
Thymeleafファイル

- Thymeleafファイルは、src/main/resources/templates内に配置する
  - 拡張子は、**.html** にする



コントローラーとThymeleafの連携

- コントローラーからThymeleafにフォワードをする際には、Modelを経由することで、データを渡すことができる



Modelへの格納とフォワード

- コントローラー側では、Model#addAttribute()メソッドを使い、Modelにデータを格納し、フォワードを行う
  - 第一引数：Thymeleafから参照する際の名前
  - 第二引数：格納するデータ

```
@GetMapping("/item")
public String showItem(Model model) {
  // Modelへの格納
  model.addAttribute("name", "りんご");
  model.addAttribute("price", 100);

  // フォワード
  return "itemPage";
}
```

Modelからのデータ取得

- Thymeleaf側では**${ }**を使い、Modelに格納されたものを参照する

```
<h2 th:text="${name}"><h2>
<p>価格: [[${price}]]円</p>
```

th:〇〇属性の外では、
${} を **[[ ]]** で囲む必要がある

- ${ }を使い表現されるものは、**EL式**と呼ばれ、簡単なプログラムを埋め込むことができる
  - Springでは、EL式を拡張した**SpEL**(**Sp**ring **E**xpression **L**anguage)が採用されている

```
<p>[[${price > 10000 ? '送料無料' :  '送料がかかります'}]]</p>
```

複数データの受け渡し

- 複数のデータを一つのオブジェクトにまとめて、コントローラーからThymeleafに渡すことも可能
  - 関連する複数のデータを取りまとめ、受け渡しのために利用されるオブジェクトを**DTO**(Data Transfer Object)と呼ぶ

例：商品名と値段のデータをItemオブジェクトとしてまとめる

```
// Modelへの格納
model.addAttribute("item", new Item("りんご", 100));

// フォワード
return "itemPage";
```

複数データの受け渡し

- DTOクラスにはGetterメソッドが必要になる
  - メソッド名は「get + フィールド名」にする

例：前頁で登場するItemクラス

```
public class Item {

  private String name;
  private int price;

  public Item(String name, int price) {
     this.name = name;
     this.price = price;
  }

     public String getName() {
          return name;
     }

     public int getPrice() {
          return price;
     }

}
```

GetterメソッドはLombokで作成してもよい

複数データの受け渡し

- EL式内では、「ドット記号 + フィールド名」を使い、データにアクセスする
  - .〇〇と記述することで、get〇〇メソッドが呼び出される

Thymeleafの記述例

```
<h1>商品情報</h1>
<ul>
    <li>商品名： [[${item.name}]]</li>
    <li>価格： [[${item.price}]]円</li>
</ul>
```

getName() が呼び出される

getPrice() が呼び出される

Null条件演算子 (Safe Navigation Operator)

- 以下の例では、itemがnullの場合に、例外が発生する

nullの場合に例外発生

```
<ul>
    <li>商品名： [[${item.name}]]</li>
    <li>価格： [[${item.price}]]円</li>
</ul>
```

- Null条件演算子「?」を付けることで、例外は発生しなくなる

```
<ul>
    <li>商品名： [[${item?.name}]]</li>
    <li>価格： [[${item?.price}]]円</li>
</ul>
```

配列, List, Mapの参照

- Thymeleafで配列やList、Mapオブジェクトを参照する場合は[ ]でインデックス番号やキーを指定する
  - 配列やListオブジェクトの参照

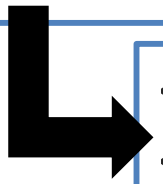
```
${配列ブジェクト名[インデックス番号]}
```

  - Mapオブジェクト参照

```
${Mapオブジェクト名["キー"]}
```

例：配列オブジェクトの出力

```
String[] items = {"りんご", "みかん"};
model.addAttribute("items", items);
```



```
<li>[[${items[0]}]]</li>
<li>[[${items[1]}]]</li>
```

Thymeleaf特有のタグや属性

- Thymeleaf内では、「th:〇〇」という特有のタグや属性を使うことができる
  - <th:block>タグやth:text属性などが存在する
  - th:〇〇属性の値として、EL式を記述することができる

Thymeleaf特有のタグや属性の記述例

```
<th:block th:each="info : ${infoList}">
    <h2 th:text="${info.title}"></h2>
    <p th:text="${info.message}"></p>
</th:block>
```

Thymeleaf特有のタグや属性については、後の章で詳しく学習する

練習問題

- 練習05-1

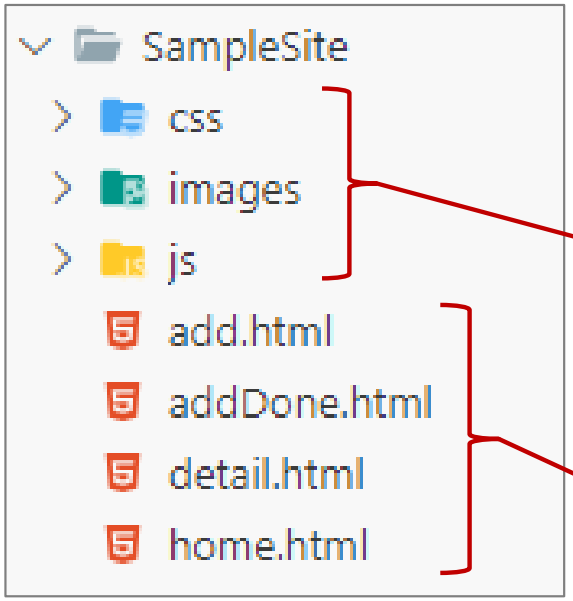
参考：静的サイトからの作業手順

作業手順の例

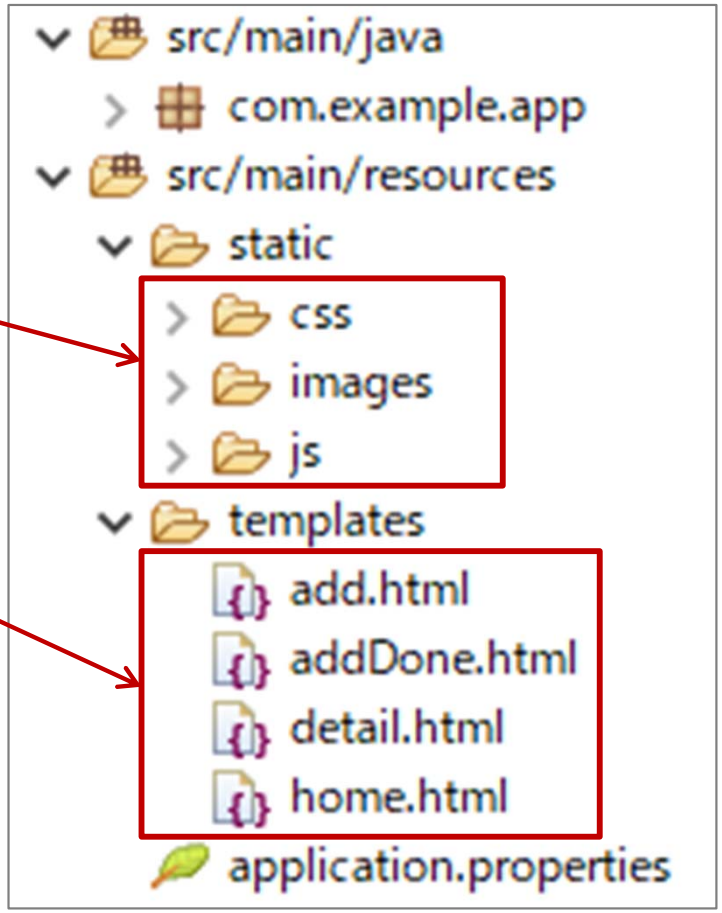
1. HTML/CSS/JavaScriptを完成させる(静的サイトとして完成)
2. EclipseでSpringのWebプロジェクトを作成する
3. CSS/JavaScript/画像フォルダを src/main/resources/static内に配置する
4. HTMLをsrc/main/resources/templates内に配置する
5. コントローラーを作成し、URLとHTMLとマッピングする
6. 各ページへのリンクや静的ファイルへのリンクを修正する
7. フォーム処理やデータベース連携といったJavaのプログラムに取り組み、必要に応じてThymeleaf(HTML)ファイルも修正する

3-4. フォルダ・ファイルの配置

静的サイト



Spring Webプロジェクト



5. コントローラー・HTMLのマッピング

- コントローラーを作成し、URLとHTMLとマッピングする

```
@Controller
public class HomeController {
    @GetMapping({"/", "/home"})
    public String showHome() {
        return "home";
    }
}
```

```
@Controller
@RequestMapping("/items")
public class ItemController {
    @GetMapping("/show")
    public String showItem() {
        return "detail";
    }

    @GetMapping("/register")
    public String registerItem() {
        return "add";
    }
    ...略
}
```

6. 各ページへのリンク修正

- a要素のhref属性を修正する
  - リンク先のURLは、コントローラーで設定した各ページのURLに合わせ、Thymeleafの提供する**th:href**属性を使い、絶対パスで指定する
    ⇒ th:href属性の値は**@{ }**で囲む形で記述する

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
…
<ul>
  <li><a th:href="@{/index}">ホーム</a></li>
  <li><a th:href="@{/items/show}">備品リスト</a></li>
  <li><a th:href="@{/items/add}">備品登録</a></li>
</ul>
…
```

th:href属性を使うために必要

コントローラーで設定したURL

6. 静的ファイルへのリンク修正

- CSS, JS, 画像といった静的ファイルへのリンクは、th:hrefやth:srcといったThymeleafの属性を使い、絶対パスで指定する
  - 通常のhrefやsrc属性を使い、相対パスでリンクを張ることも可能だが、コントローラーで指定しているURLとの兼ね合いでリンク切れを起こしてしまうことがある

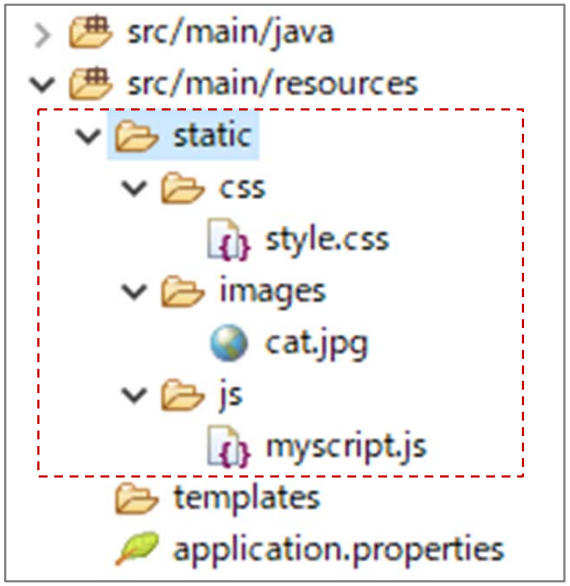
例：右図の静的ファイルへのリンク

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

```
<img th:src="@{/images/cat.jpg}">
```

```
<script th:src="@{/js/myscript.js}"></script>
```

※「static」というフォルダ名は記述しない



練習問題

- 練習05-2