フレームワーク基礎実習

08. Thymeleaf 基礎

株式会社ジードライブ

今回学ぶこと

- Thymeleafのタグや属性
  - 変数の出力
  - 分岐処理と反復処理
  - 共通要素のパーツ化

Thymeleafをフォームで利用する方法については、
後の章で学習する

変数の出力

- [[　]] 内に${…}で記述する

```
<h1>[[${title}]]</h1>
<ul>
    <li>名前：[[${user.name}]]</li>
    <li>年齢：[[${user.age}]]歳</li>
</ul>
```

getName() が呼び出される

- th:text属性を使用する

```
<h1 th:text="${title}"></h1>
<ul>
    <li th:text="'名前：' + ${user.name}"></li>
    <li th:text="|年齢：${user.age} 歳|"></li>
</ul>
```

+ による文字列結合
文字列はシングルクォートで囲む

パイプ（縦線）で囲むことで、簡易的に文字列結合ができる

※ 出力する文字列にHTMLタグが含まれる場合は、**utext**を使用

ローカル変数の定義

- ローカル変数は th:with属性 として設定できる
  - この変数は子要素内で参照することができる

```
<ul th:with="name='山田太郎', age=25">
    <li th:text="'氏名：' + ${name}"></li>
    <li th:text="|年齢：${age}歳|"></li>
</ul>
```

th:object / th:block

- 親要素で th:object属性 として設定したものは、子要素では ＊(アスタリスク) で参照することができる

```
<ul th:object="${user}">
    <li th:text="'氏名：' + *{name}"></li>
    <li th:text="|年齢：*{age}歳|"></li>
</ul>
```

${user.age} と同等

- th:block はHTML変換時に削除されるタグで、様々な要素の親要素として使用することができる

```
<th:block th:object="${user}">
    <p th:text="'氏名：' + *{name}"><p>
    <p th:text="|年齢：*{age}歳|"></p>
</th:block>
```

param / session

- GETパラメータやPOSTパラメータは ${param.〇〇} で参照することができる

```
<p th:text="${param.name}"></p>
<p th:text="${param.age}"></p>
```

  - 分岐処理等で使用する場合には、**${param.name[0]}**のように記述する

- セッションに格納されているデータは ${session.〇〇} で参照することができる

```
<p th:text="${session.name}"></p>
<p th:text="${session.age}"></p>
```

パスの出力

- コンテキストルートを含む絶対パスを出力したい場合は、@{…} の形式で出力する
  - th:href属性 や th:src属性の値として記述することが一般的

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

<a th:href="@{/item/list}"></a>

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

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

パスの出力

- URLにパラメーターを含む場合、以下のように記述する

例：localhost:8080/list**?area=3&page=5** のような形式で出力する

```
<a th:href="@{/list(area=${areaId},page=${pageNum})}">
```

例：localhost:8080/list**/3/5** のような形式で出力する

```
<a th:href=
"@{/list/{area}/{page}(area=${areaId}, page=${pageNum})}">
```

- 画像ファイル名を変数にする場合は、以下のように記述する

```
<img th:src="@{/images/{name}(name=${photo})}">
```

フォーマット

- 数値や日時のフォーマットを行う場合は、#numbersや#datesといったThymeleafで利用可能な暗黙オブジェクトのメソッドを使用する

例：整数を3桁区切りにする(#numbersオブジェクトを利用)

コントローラー

```
model.addAttribute("price", 123456789);
```

```
<p th:text=
     "${#numbers.formatInteger(price, 3, 'COMMA')}"></p>
```

フォーマット

例：java.util.Dateのフォーマット(#datesオブジェクトを利用)

コントローラー

```
model.addAttribute("created", new Date());
```

```
<p th:text=
        "${#dates.format(created, 'yyyy年MM月dd日')}"></p>
```

例：java.timeパッケージの場合(#temporalsオブジェクトを利用)

コントローラー

```
model.addAttribute("created", LocalDateTime.now());
```

```
<p th:text=
        "${#temporals.format(created, 'yyyy年MM月dd日')}"></p>
```

カスタムデータ属性

- カスタムデータ属性は、HTMLにおいて開発者が自由に定義できる属性
  - 属性名の頭に data- を付ける必要がある
  - JavaScriptと組み合わせて利用される

カスタムデータ属性の例

```
<li data-id="1" data-address="東京都">山田太郎</li>
```

- カスタムデータ属性の値として、変数を使用したい場合は、th:data-〇〇属性を使用する

```
<li th:data-id="${user.id}"
     th:data-address="${user.address}">山田太郎</li>
```

練習

- 練習08-1
- 練習08-2

分岐処理と反復処理

th:if / th:unless

- 単一分岐の場合は、th:if または th:unless を使用する

```
<p th:if="${price >= 10000}">送料無料です</p>
```

  - 上記の例をunlessで記述すると以下のようになる

```
<p th:unless="${price < 10000}">送料無料です</p>
```

- else if / else は存在しない (以下のように記述して対応する)

```
<p th:if="${age >= 30}">衆参議員に立候補可</p>
<p th:if="${age >= 25 && age < 30}">衆議員に立候補可</p>
<p th:unless="${age >= 25}">議員に立候補不可</p>
```

- 文字列比較はequalsメソッドで行う

```
<p th:if="${name.equals('山田')}"> … </p>
```

th:switch

- 複数に分岐させる場合、th:switchを利用できる
- 条件は th:case属性で記述する
  - ＊(アスタリスク)で、その他を表現することができる

例：セッションに格納されている言語設定に応じて挨拶を切り替える

```
<th:block th:switch="${session.lang}">
        <p th:case="ja">おはようございます</p>
        <p th:case="es">Buenos días</p>
        <p th:case="it">Buongiorno</p>
        <p th:case="*">Good morning</p>
</th:block>
```

th:each

- th:each属性を使用することで、拡張for文のように配列やリストを扱うことができる

```
<table>
    <tr>
        <th>名前</th>
        <th>年齢</th>
    </tr>
    <tr th:each="user : ${userList}">
        <td>[[${user.name}]]</td>
        <td>[[${user.age}]]歳</td>
    </tr>
</table>
```

ループ内で使用する変数

配列またはリスト

このブロックが繰り返される

th:each

- ステータス用の変数を用意することで、ループ内の状態を知ることができる

変数名は任意

```
<tr th:each="user, status : ${userList}">
    <td>[[${status.index}]]</td>
    <td>[[${user.name}]]</td>
    <td>[[${user.age}]]歳</td>
</tr>
```

変数のもつプロパティ	説明
index / count	周回数。indexは0始まり
size	配列やリストの要素数
odd / even	奇数番目 / 偶数番目 か否かを真偽値で返す
first/last	最初 / 最後 か否かを真偽値で返す

th:each

- th:eachとth:objectを併用することもできる

```
<tr th:each="user : ${userList}" th:object="${user}">
    <td>[[*{name}]]</td>
    <td>[[*{age}]]歳</td>
</tr>
```

$ の代わりに ***** が使用できるようになり、
user. を省略できる

th:each

- 繰り返しの最初の数と最後の数が決まっている場合は、#numbers.sequence()を利用する

```
<table>
    <caption>九九の表</caption>
    <tr th:each="i : ${#numbers.sequence(1, 9)}">
        <th>[[${i}]]の段</th>
        <td th:each="j : ${#numbers.sequence(1, 9)}">
            [[${i * j}]]
        </td>
    </tr>
</table>
```

練習

- 練習08-3

共通要素のパーツ化

th:fragment

- ヘッダーやフッターといった複数ページで共通する要素をパーツ化したものをフラグメントと呼ぶ

```
<!doctype html>
<html lang="ja" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8">
</head>
<body>
<div th:fragment="page_header">
    <h1>[[${title}]]</h1>
    <ul><li><a th:href="@{/about}">About</a></li>
        <li><a th:href="@{/contact}">Contact</a></li></ul>
</div>
<th:block th:fragment="page_footer">
    <footer><p><small>&copy; Taro Yamada</small></p></footer>
    <script th:src="@{/js/script.js}"></script>
</th:block>
</body>
</html>
```

「ファイル名::page_header」で読み込まれる

「ファイル名::page_footer」で読み込まれる

th:insert / th:replace

- フラグメントは、th:insert や th:replace属性で読み込むことができる
  - ~{ファイル名 :: フラグメント名}で読み込むフラグメントを指定

```
<!doctype html>
<html lang="ja" xmlns:th="http://www.thymeleaf.org">
<head th:fragment="html_head">
<meta charset="UTF-8">
</head>
<body>
<header th:insert="~{common :: page_header}"></header>
<main>
   <h1>メインコンテンツ</h1>
</main>
<div th:replace="~{common :: page_footer}"></div>
</body>
</html>
```

このheaderタグの**中に**、読み込まれた要素が配置される

common.html の page_header を読み込む

このdivタグの**代わりに**、読み込まれた要素が配置される

common.html の page_footer を読み込む

th:insert / th:replace

- フラグメント化されていなくても、タグ名、class属性、id属性を使い、読み込むことも可能

読み込まれる要素 (common.html)

```
<h1 id="site_name" class="site_name" th:remove="tag">
    <span>ブログ</span>My Blog
</h1>
```

読み込み先で、このh1タグは消える

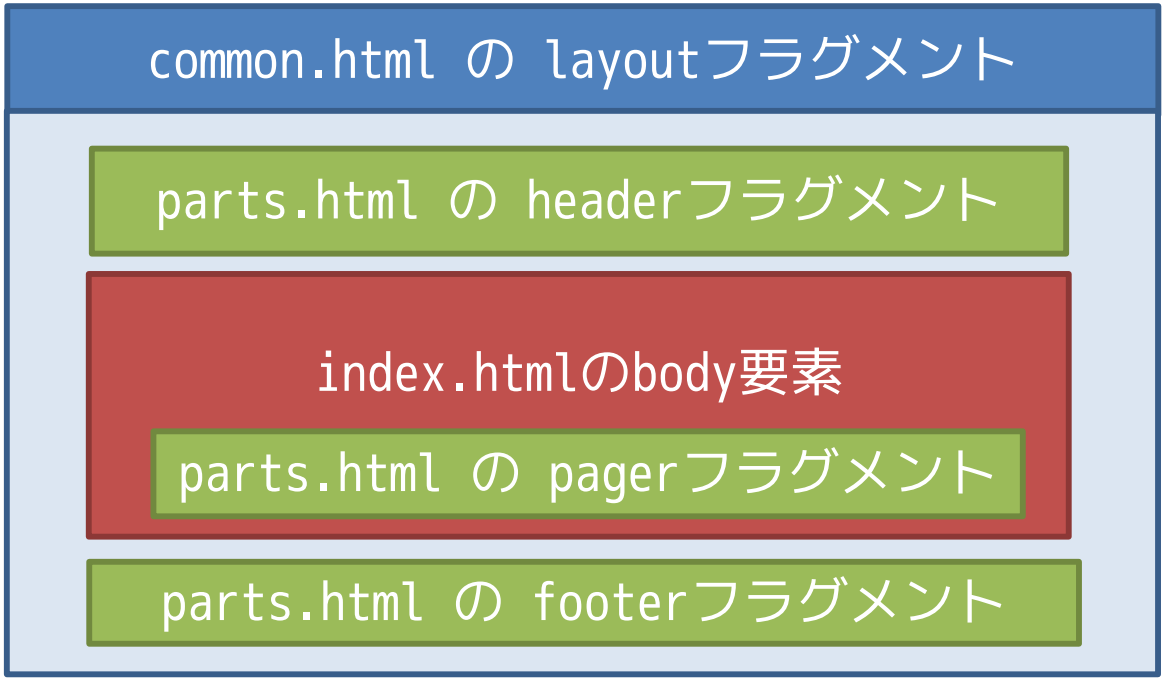
以下は、基本的にどれも同じ結果になる

```
<div th:insert="~{common :: h1}"></div>
<div th:insert="~{common :: .site_name}"></div>
<div th:insert="~{common :: #site_name}"></div>
```

※ h1や.site_nameが複数存在する場合は、すべて読み込まれる
※ ファイル名を省略し、{:: h1} のように記述すると、自ファイルを指す

フラグメントの応用

- フラグメントには引数の設定が可能で、それにより、パーツに汎用性をもたせることができる
  - 以降のページでは、index.htmlが呼び出された際に、以下のようなレイアウトでHTMLを出力する方法を紹介する



parts.html

- parts.htmlはヘッダー、フッター、ページ番号のフラグメントをもつ

```
<!doctype html>
<html lang="ja" xmlns:th="http://www.thymeleaf.org">
<body>
<header th:fragment="header">
    <h1>Taro's Blog</h1>
</header>

<footer th:fragment="footer">
    <p><small>&copy; Taro Yamada</small></p>
</footer>

<div th:fragment="pager">
    <p><a href="">1</a><a href="">2</a><a href="">3</a></p>
</div>
</body>
</html>
```

ヘッダー

フッター

ページ番号

common.html

- common.htmlはヘッダー、フッター、及びメインコンテンツをもつ
  - html全体がlayoutというフラグメント名になっており、引数を２つ取る

```
<!doctype html>
<html lang="ja" xmlns:th="http://www.thymeleaf.org"
        th:fragment="layout(main, title)">
<head th:fragment="html_head">
<meta charset="UTF-8">
<title>[[${title}]]</title>
<link rel="stylesheet" th:href="@{/css/style.css}">
</head>
<body>
<div th:replace="~{parts :: header}"></div>

<div th:replace="${main}"></div>

<div th:replace="~{parts :: footer}"></div>
</body>
</html>
```

ヘッダーの読み込み

メインコンテンツ：
index.htmlのbody要素が読み込まれる

フッターの読み込み

index.html

- index.htmlは、htmlタグにth:replace属性が付与されており、要素全体がcommon.htmlのlayoutフラグメントに置き換わる

```
<!doctype html>
<html lang="ja" xmlns:th="http://www.thymeleaf.org"
    th:replace="~{common :: layout(~{::body/content()}, 'トップページ')}">
<body>
<article>
    <h2>荻窪散歩</h2>
    <p>今日は久しぶりに荻窪の駅前商店街を...</p>
</article>
<article>
    <h2>しながわ水族館</h2>
    <p>品川駅近くにある水族館に初めて...</p>
</article>

<div th:replace="~{parts :: pager}"></div>
</body>
</html>
```

layoutフラグメントを呼び出すにあたって、body要素内のコンテンツを引数として渡している

ページ番号の読み込み

練習

- 練習08-4