

Spring Boot 基本學習

Introduction

本資料はSpring Boot 学習者のための、Spring Bootの基本的な挙動を把握するための学習資料である。

1～4章で成り立っており、それぞれ、実際のコードを理解しつつ、実行して確認するという流れになっている。

注1) 本資料は「Springプラグイン導入手順書」を読んでいることを前提とする。

注2) ビルドツールは、Java開発において多く使われているGradleを使用する。

目次

1, コントローラーの作成

1-1, プロジェクトの作成

1-2, 学習ポイント1

1-3, 学習ポイント2

1-4, サンプル実行

2, Thymeleafを用いたロジック

2-1, プロジェクトの作成

2-2, 学習ポイント1

2-3, 学習ポイント2

2-4, サンプル実行

目次

3, 複数値の送信

3-1, プロジェクトの作成

3-2, 学習ポイント1

3-3, 学習ポイント2

3-4, サンプル実行

4, Thymeleafのeach文

4-1, プロジェクトの作成

4-2, 学習ポイント1

4-3, 学習ポイント2

4-4, サンプル実行

1, コントローラーの作成

1, コントローラーの作成

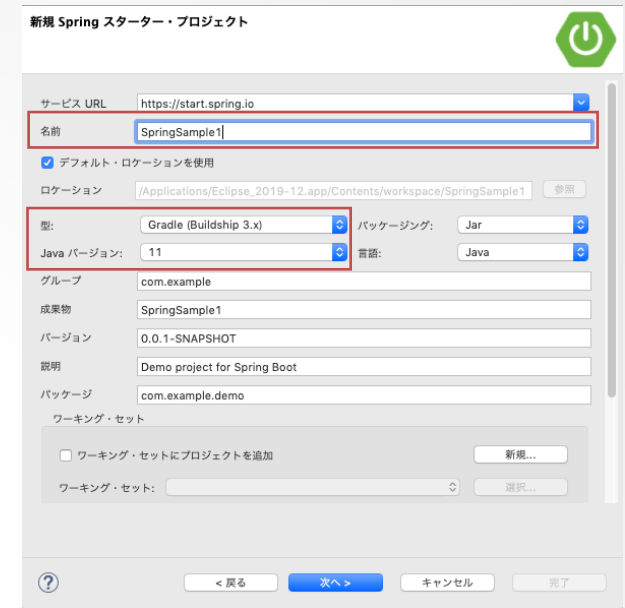
1-1, プロジェクトの作成

1, 新規 Spring スターター・プロジェクト

名前: SpringSample1

型: Gradle (Buildship 3.x)

Java バージョン: 11



新規 Spring スターター・プロジェクト

サービス URL:

名前:

☒ デフォルト・ロケーションを使用

ロケーション:

型: バックエンド:

Java バージョン: 言語:

グループ:

成果物:

バージョン:

説明:

パッケージ:

ワーキング・セット

☐ ワーキング・セットにプロジェクトを追加

ワーキング・セット:

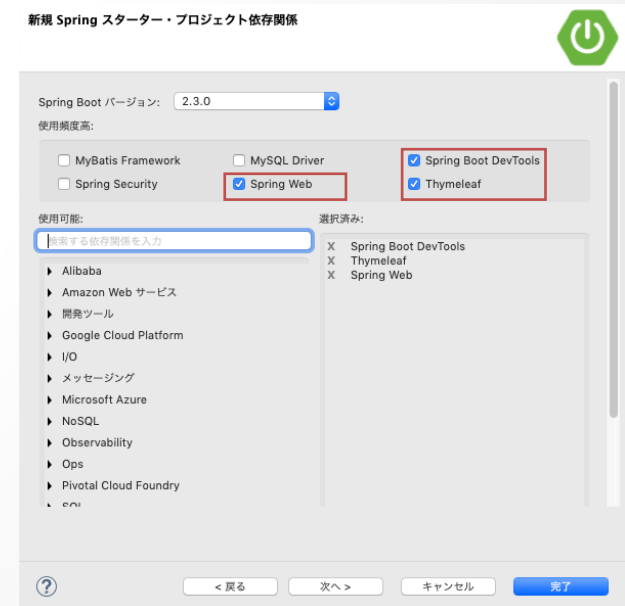
< 戻る 次へ > キャンセル 完了

2, 新規 Spring スターター・プロジェクト依存関係

下記の項目をチェック。

- Spring Web
- Thymeleaf
- Spring Boot DevTools

※画面上部に表示されていない場合は、検索欄を使用。



新規 Spring スターター・プロジェクト依存関係

Spring Boot バージョン:

使用頻度高:

☐ MyBatis Framework ☐ MySQL Driver ☒ Spring Boot DevTools

☐ Spring Security ☒ Spring Web ☒ Thymeleaf

使用可能:

検索する依存関係を入力

Alibaba
Amazon Web サービス
開発ツール
Google Cloud Platform
I/O
メッセージング
Microsoft Azure
NoSQL
Observability
Ops
Pivotal Cloud Foundry

選択済み:

X Spring Boot DevTools
X Thymeleaf
X Spring Web

< 戻る 次へ > キャンセル 完了

1, コントローラーの作成

1-1, プロジェクトの作成

3, 追加ファイル

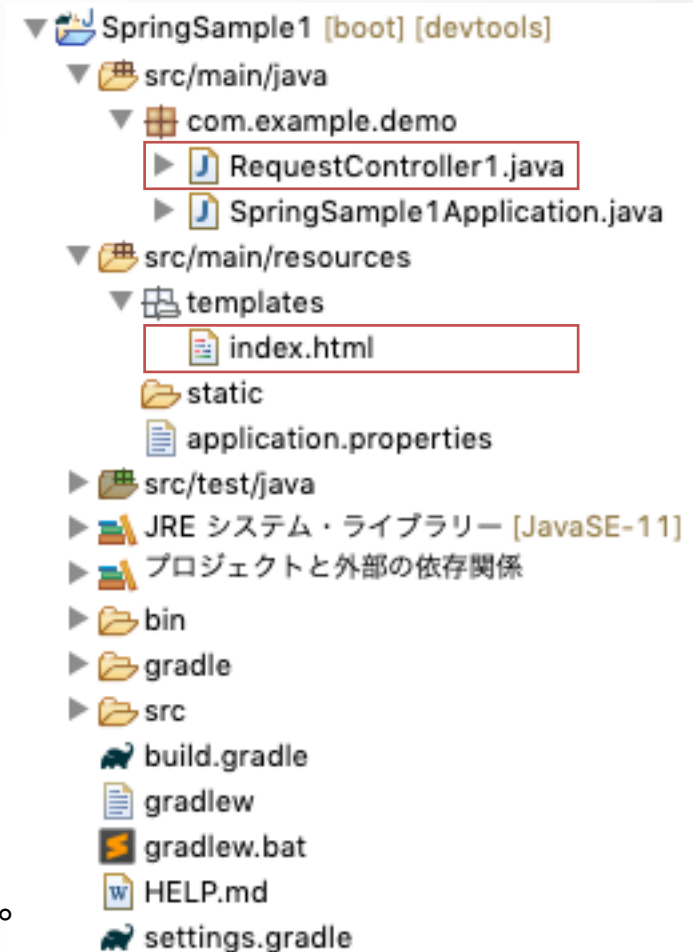
Javaクラスファイル

RequestController1.java

HTMLファイル

index.html

※ファイル内容は、サンプルファイルよりコピー&ペースト。



1, コントローラーの作成

1-2, 学習ポイント1

アノテーション

@を先頭にした文字列。

Springでは、アノテーションにより特定の役割を与えることが出来る。

コントローラー

MVC(Model View Controller)における、Controllerのことである。
ModelとViewを、橋渡しするのが役割。

1, コントローラーの作成

1-2, 学習ポイント1

RequestController1.java

```
import org.springframework.stereotype.Controller;
```

```
@Controller
```

```
public class RequestController1 {
```

```
}
```



RequestProcクラスに@Controllerアノテーションを付けている。
それにより、Spring FrameworkにRequestProcクラスを、
コントローラーだと認識させている。

1, コントローラーの作成

1-3, 学習ポイント2

関数の引数にアノテーションを指定

引数にアノテーションを指定することが出来る。

指定したアノテーションに応じた値を、引数として受け取ることが出来る。

1, コントローラーの作成

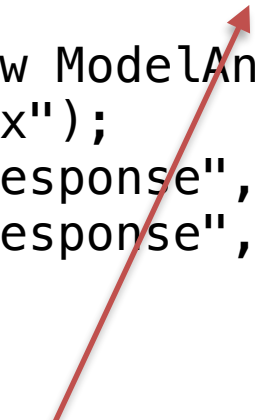
1-3, 学習ポイント2

RequestController1.java

```
@RequestMapping(value="/callResponse")
public ModelAndView response(@RequestParam("name") String name,
                             @RequestParam("tweet") String tweet) {

    ModelAndView mav = new ModelAndView();
    mav.setViewName("index");
    mav.addObject("helloResponse", "Hello. I am " + name + ".");
    mav.addObject("tweetResponse", tweet);

    return mav;
}
```



response関数の引数に、@RequestParamアノテーションを付けた引数を渡している。
それにより、URLに含まれる文字列などのパラメータを、関数の引数に渡すことが出来る。

1, コントローラーの作成

1-4, サンプル実行

1, SpringSample1 を実行する。



2, <http://localhost:8080/> にアクセスする。

3, NameとTweetの欄に、文字を入れる。

4, Sendボタンを押下。

5, 画面下部に、文字が送信される。

A screenshot of a web browser at localhost:8080. The page title is 'Tweet Message'. It contains two input fields: 'Name' with the value 'Mike' and 'Tweet' with the value 'How are you?'. Below the input fields is a 'Send' button.

← → ↻ ⓘ localhost:8080

Tweet Message

Name

Tweet

A screenshot of a web browser at localhost:8080/callResponse?name=Mike&tweet=How+are+you%3F. The page title is 'Tweet Message'. It contains two input fields: 'Name' and 'Tweet'. Below the input fields is a 'Send' button. Below the 'Send' button, the text 'Hello. I am Mike.' and 'How are you?' is displayed.

← → ↻ ⓘ localhost:8080/callResponse?name=Mike&tweet=How+are+you%3F

Tweet Message

Name

Tweet

Hello. I am Mike.

How are you?

2, Thymeleafを用いたロジック

2, Thymeleafを用いたロジック

2-1, プロジェクトの作成

1, Spring スターター・プロジェクト作成

名前 : SpringSample2

型 : Gradle (Buildship 3.x)

Java バージョン : 11

新規 Spring スターター・プロジェクト

サービス URL: <https://start.spring.io>

名前: SpringSample2

☒ デフォルト・ロケーションを使用

ロケーション: /Applications/Eclipse_2019-12.app/Contents/workspace/SpringSample2

型: Gradle (Buildship 3.x)

Java バージョン: 11

パッケージング: Jar

言語: Java

グループ: com.example

成果物: SpringSample2

バージョン: 0.0.1-SNAPSHOT

説明: Demo project for Spring Boot

パッケージ: com.example.demo

ワーキング・セット

☐ ワーキング・セットにプロジェクトを追加

ワーキング・セット:

< 戻る 次へ > キャンセル 完了

2, 新規 Spring スターター・プロジェクト依存関係

下記の項目をチェック。

- Spring Web
- Thymeleaf
- Spring Boot DevTools

※画面上部に表示されていない場合は、検索欄を使用。

新規 Spring スターター・プロジェクト依存関係

Spring Boot バージョン: 2.3.0

使用頻度高:

☐ MyBatis Framework ☐ MySQL Driver ☒ Spring Boot DevTools

☐ Spring Security ☒ Spring Web ☒ Thymeleaf

使用可能:

検索する依存関係を入力

Alibaba

Amazon Web サービス

開発ツール

Google Cloud Platform

I/O

メッセージング

Microsoft Azure

NoSQL

Observability

Ops

Pivotal Cloud Foundry

選択済み:

X Spring Boot DevTools

X Thymeleaf

X Spring Web

< 戻る 次へ > キャンセル 完了

2, Thymeleafを用いたロジック

2-1, プロジェクトの作成

3, 追加ファイル

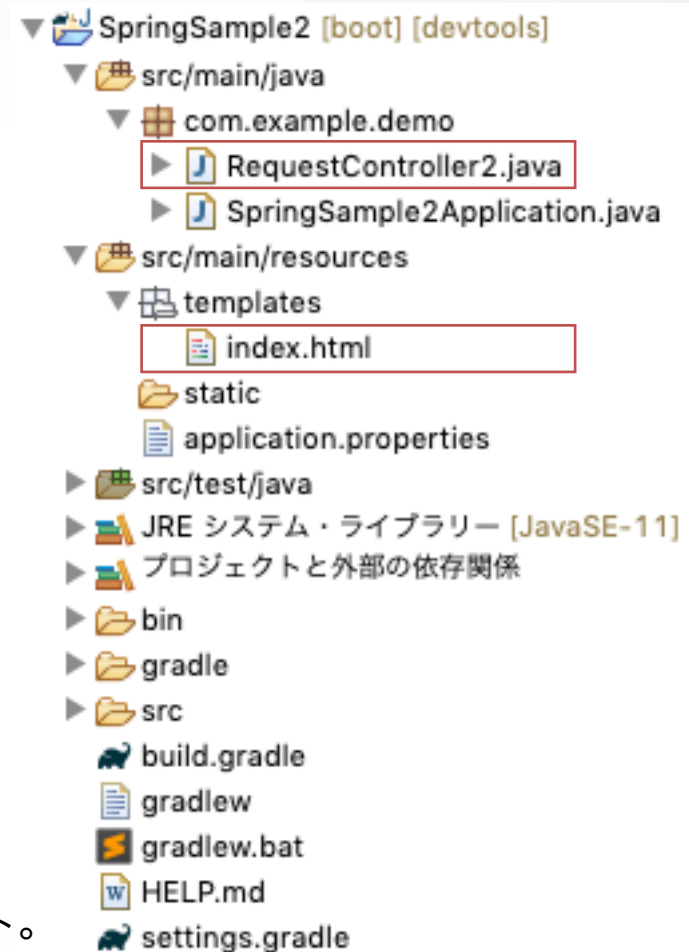
Javaクラスファイル

RequestController2.java

HTMLファイル

index.html

※ファイル内容は、サンプルファイルよりコピー&ペースト。



2, Thymeleafを用いたロジック

2-2, 学習ポイント1

Thymeleafを用いたHTML内のロジック

Thymeleafとは、HTML内にThymeleafのタグを入れることにより使用できるテンプレートエンジン。

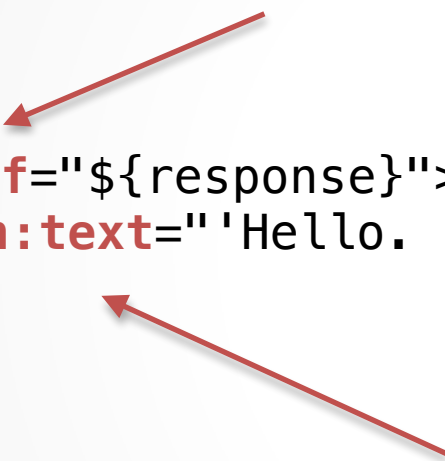
単純な表示機能に加えて、HTML内にロジックを入れ込むことを可能にする。

2, Thymeleafを用いたロジック

2-2, 学習ポイント1

index.html

変数responseに値が入っている場合はTrueと判断され、このタグと内部タグが処理される。
responseに値が入っていない場合はFalseと判断されて、処理は行われない。



```
<div th:if="${response}">  
  <h2 th:text="'Hello. I am ' + ${response.name} + '.'"></h2>  
</div>
```

th:textは、文字列の表示、変数内の表示に加えて、
文字列結合も可能。

2, Thymeleafを用いたロジック

2-3, 学習ポイント2

リクエストパラメータで、Javaオブジェクトを受け取る

リクエストパラメータで、引数1つに対して1つの値を受け取ることは、値の個数が増えることを想定すると、引数の数が増大する原因になる。よって、値をJavaオブジェクトに格納してからリクエストパラメータに渡すことにより、引数1つで複数の値を受け取ることが出来る。

2, Thymeleafを用いたロジック

2-3, 学習ポイント2

RequestController2.java

```
@RequestMapping(value="/callResponse")
public ModelAndView response(@ModelAttribute ParamObject param) {

    ModelAndView mav = new ModelAndView();
    mav.setViewName("index");
    mav.addObject("response", param);
    return mav;
}
```

```
public static class ParamObject {

    private String name;
    private String tweet;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getTweet() { return tweet; }
    public void setTweet(String tweet) { this.tweet = tweet; }
}
```

Javaオブジェクトを受け取ることにより、
引数が1つになっている。

メンバー変数nameとtweetを持ったクラス。

nameとtweetの値を格納するために使用する。値が増えた場合は、適宜増やしていく。

2, Thymeleafを用いたロジック

2-4, サンプル実行

1, SpringSample2を実行する。



2, <http://localhost:8080/> にアクセスする。

3, NameとTweetの欄に、文字を入れる。

4, Sendボタンを押下。

5, 画面下部に、文字が送信される。

A screenshot of a web browser showing a form titled 'Tweet Message'. The form has two input fields: 'Name' with the value 'Mike' and 'Tweet' with the value 'How are you?'. Below the fields is a 'Send' button. The browser's address bar shows 'localhost:8080'.A screenshot of a web browser showing the result of the form submission. The browser's address bar shows 'localhost:8080/callResponse?name=Mike&tweet=How+are+you%3F'. The page displays the text 'Hello. I am Mike.' and 'How are you?' below the 'Tweet Message' title. The form fields and 'Send' button are no longer visible.

3, 複数値の送信

3, 複数値の送信

3-1, プロジェクトの作成

1, Spring スターター・プロジェクト作成

名前: SpringSample3

型: Gradle (Buildship 3.x)

Java バージョン: 11

新規 Spring スターター・プロジェクト

サービス URL: <https://start.spring.io>

名前: SpringSample3

☒ デフォルト・ロケーションを使用

ロケーション: [Applications/Eclipse_2019-12.app/Contents/workspace/SpringSample3](#) 参照

型: Gradle (Buildship 3.x) パッケージング: Jar

Java バージョン: 11 言語: Java

グループ: com.example

成果物: SpringSample3

バージョン: 0.0.1-SNAPSHOT

説明: Demo project for Spring Boot

パッケージ: com.example.demo

ワーキング・セット

☐ ワーキング・セットにプロジェクトを追加

ワーキング・セット: 選択...

< 戻る 次へ > キャンセル 完了

2, 新規 Spring スターター・プロジェクト依存関係

下記の項目をチェック。

- Spring Web
- Thymeleaf
- Spring Boot DevTools

※画面上部に表示されていない場合は、検索欄を使用。

新規 Spring スターター・プロジェクト依存関係

Spring Boot バージョン: 2.3.0

使用頻度高:

☐ MyBatis Framework ☐ MySQL Driver ☒ Spring Boot DevTools

☐ Spring Security ☒ Spring Web ☒ Thymeleaf

使用可能:

検索する依存関係を入力

選択済み:

- X Spring Boot DevTools
- X Thymeleaf
- X Spring Web

Alibaba Amazon Web サービス 開発ツール Google Cloud Platform I/O メッセージング Microsoft Azure NoSQL Observability Ops Pivotal Cloud Foundry

< 戻る 次へ > キャンセル 完了

3, 複数値の送信

3-1, プロジェクトの作成

3, 追加ファイル

Javaクラスファイル

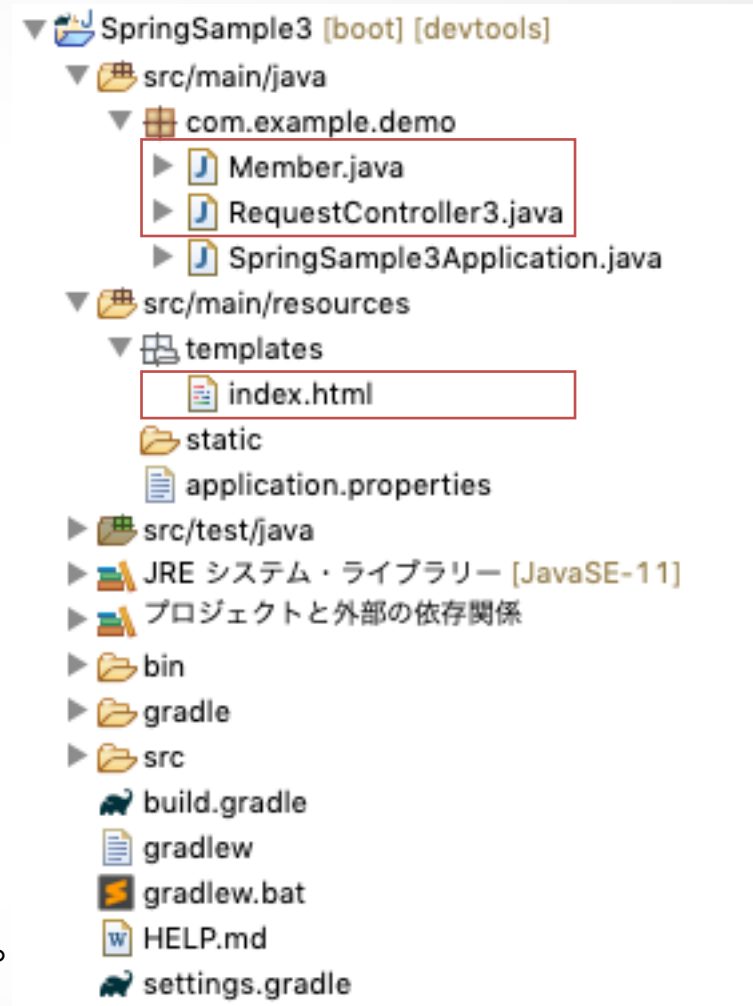
Member.java

RequestController3.java

HTMLファイル

index.html

※ファイル内容は、サンプルファイルよりコピー&ペースト。



3, 複数値の送信

3-2, 学習ポイント1

複数のテキストボックスにおける番号の指定

入力項目が複数になった場合には、どの項目に何の値が入っているかを、明確化する必要がある。

そのため、添え字を用いてテキストボックスに番号を割り振る。

3, 複数値の送信

3-2, 学習ポイント1

index.html

変数の前に、members[*].で添え字を付けることにより、テキストボックスの位置を明確化している。

```
<tr>
  <td><input type="text" name="members[0].name"></td>
  <td><input type="text" name="members[0].address"></td>
  <td><input type="text" name="members[0].telno"></td>
</tr>
<tr>
  <td><input type="text" name="members[1].name"></td>
  <td><input type="text" name="members[1].address"></td>
  <td><input type="text" name="members[1].telno"></td>
</tr>
<tr>
  <td><input type="text" name="members[2].name"></td>
  <td><input type="text" name="members[2].address"></td>
  <td><input type="text" name="members[2].telno"></td>
</tr>
```

<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>

← members[0]の行
← members[1]の行
← members[2]の行

※表示例

3, 複数値の送信

3-3, 学習ポイント2

引数でリスト値のオブジェクトを受け取る

表形式のデータを受け取る際に、行が増加する場合が想定される。
そのため、1行ごとにリストに格納する方法が望ましい。
そうすることにより、表の行数が増えた場合でも、1つのオブジェクトとして受け取ることが出来る。

3, 複数値の送信

3-3, 学習ポイント2

RequestController3.java

@ModelAttribute アノテーションで、オブジェクトを受け取っている。

```
@RequestMapping(value="/post")
public ModelAndView response(@ModelAttribute ParamObject members) {

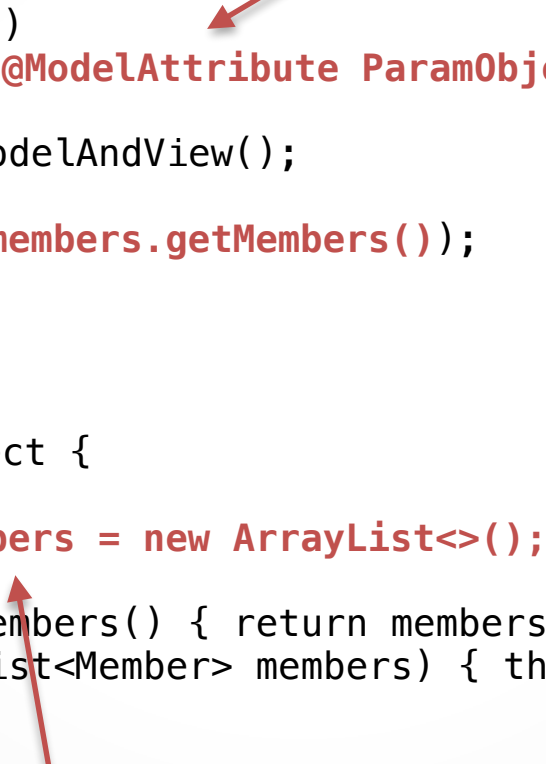
    ModelAndView mav = new ModelAndView();
    mav.setViewName("index");
    mav.addObject("result", members.getMembers());

    return mav;
}

public static class ParamObject {

    private List<Member> members = new ArrayList<>();

    public List<Member> getMembers() { return members; }
    public void setMembers(List<Member> members) { this.members = members; }
}
```



Memberクラスを型にして、リスト値のオブジェクトmembersを作成している。

3, 複数値の送信

3-4, サンプル実行

1, SpringSample3を実行する。



2, <http://localhost:8080/> にアクセスする。

3, テキストボックスに文字を入れる。

4, Sendボタンを押下。

5, 画面下部に、テキストが送信される。

← → ↻ ⓘ localhost:8080

Customer Information

Name	Address	Telno
<input type="text" value="Alex"/>	<input type="text" value="California"/>	<input type="text" value="00000"/>
<input type="text" value="Dana"/>	<input type="text" value="Florida"/>	<input type="text" value="11111"/>
<input type="text" value="Jamie"/>	<input type="text" value="Iowa"/>	<input type="text" value="22222"/>
<input type="text" value="Morgan"/>	<input type="text" value="Kansas"/>	<input type="text" value="33333"/>
<input type="text" value="Robin"/>	<input type="text" value="Idaho"/>	<input type="text" value="44444"/>
<input type="text" value="Terry"/>	<input type="text" value="Alaska"/>	<input type="text" value="55555"/>
<input type="text" value="Chris"/>	<input type="text" value="Texas"/>	<input type="text" value="66666"/>
<input type="text" value="Pat"/>	<input type="text" value="Vermont"/>	<input type="text" value="77777"/>
<input type="text" value="Tracey"/>	<input type="text" value="Wisconsin"/>	<input type="text" value="88888"/>
<input type="text" value="Ronnie"/>	<input type="text" value="Utah"/>	<input type="text" value="99999"/>



← → ↻ ⓘ localhost:8080/post?members%5B0%5D.name=Alex&members%5B0%5D.address=California&

<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>

Alex California 00000
Dana Florida 11111
Jamie Iowa 22222
Morgan Kansas 33333
Robin Idaho 44444
Terry Alaska 55555
Chris Texas 66666
Pat Vermont 77777
Tracey Wisconsin 88888
Ronnie Utah 99999

4, Thymeleafのeach文

4, Thymeleafのeach文

4-1, プロジェクトの作成

1, Spring スターター・プロジェクト作成

名前 : SpringSample4

型 : Gradle (Buildship 3.x)

Java バージョン : 11

新規 Spring スターター・プロジェクト

サービス URL: <https://start.spring.io>

名前: SpringSample4

☒ デフォルト・ロケーションを使用

ロケーション: /Applications/Eclipse_2019-12.app/Contents/workspace/SpringSample4

型: Gradle (Buildship 3.x) バッケージング: Jar

Java バージョン: 11 言語: Java

グループ: com.example

成果物: SpringSample4

バージョン: 0.0.1-SNAPSHOT

説明: Demo project for Spring Boot

パッケージ: com.example.demo

ワーキング・セット

☐ ワーキング・セットにプロジェクトを追加

ワーキング・セット:

新規...

次へ >

2, 新規 Spring スターター・プロジェクト依存関係

下記の項目をチェック。

- Spring Web
- Thymeleaf
- Spring Boot DevTools

※画面上部に表示されていない場合は、検索欄を使用。

新規 Spring スターター・プロジェクト依存関係

Spring Boot バージョン: 2.3.0

使用頻度高:

☐ MyBatis Framework ☐ MySQL Driver ☒ Spring Boot DevTools

☐ Spring Security ☒ Spring Web ☒ Thymeleaf

使用可能:

検索する依存関係を入力

Alibaba

Amazon Web サービス

開発ツール

Google Cloud Platform

I/O

メッセージング

Microsoft Azure

NoSQL

Observability

Ops

Pivotal Cloud Foundry

選択済み:

X Spring Boot DevTools

X Thymeleaf

X Spring Web

次へ >

4, Thymeleafのeach文

4-1, プロジェクトの作成

3, 追加ファイル

Javaクラスファイル

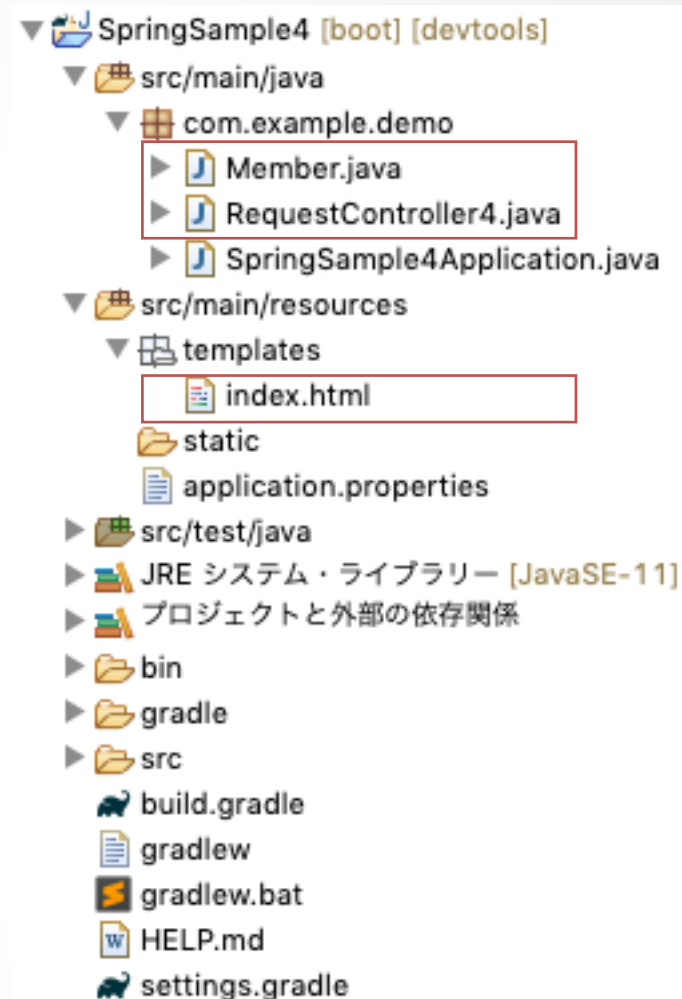
Member.java

RequestController4.java

HTMLファイル

index.html

※ファイル内容は、サンプルファイルよりコピー&ペースト。



4, Thymeleafのeach文

4-2, 学習ポイント1

HTML内での動的な値の変更

入力欄が複数になった場合に、通常であれば、その項目分のタグをHTML内に記述する必要がある。

Thymeleafを用いれば、値を動的に変更することが可能なため、冗長なコードをシンプル化することが出来る。


4, Thymeleafのeach文

4-2, 学習ポイント1

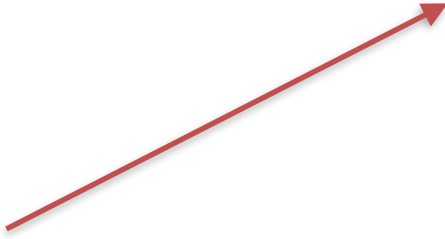
index.html

th:eachにより、ループ処理を行っている。

オブジェクトの数の分だけテキストボックスを表示している。



```
<tr th:each="member, num : ${paramObject.members}">
  <td><input type="text" th:name="'members[' + ${num.index} + '].name'"></td>
  <td><input type="text" th:name="'members[' + ${num.index} + '].address'"></td>
  <td><input type="text" th:name="'members[' + ${num.index} + '].telno'"></td>
</tr>
```



`${num.index}`を用いて、`member[*].`における『*』部分の動的な変更を行っている。

4, Thymeleafのeach文

4-3, 学習ポイント2

オブジェクトの作成タイミング

画面の初期表示時に、値を格納するためのオブジェクトを画面側(View)に、渡しておく。

そうすることにより、HTML側でループ処理が使用可能になり、必要な数だけHTMLのタグを生成出来る。

4, Thymeleafのeach文

4-3, 学習ポイント2

RequestController4.java

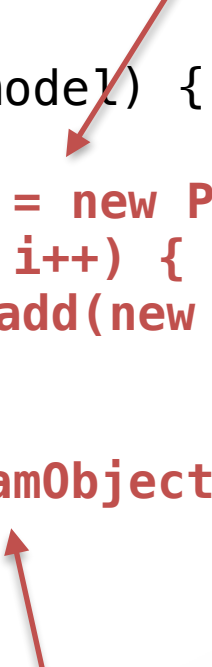
画面初期表示時に、10行分のMemberオブジェクトを生成しておく。

```
@RequestMapping(value="/")
public String index(Model model) {

    ParamObject paramObject = new ParamObject();
    for (int i = 0; i < 10; i++) {
        paramObject.members.add(new Member());
    }

    model.addAttribute("paramObject", paramObject);

    return "index";
}
```



画面側(View)に、生成したオブジェクトを渡している。

4, Thymeleafのeach文

4-4, サンプル実行

1, SpringSample4を実行する。



2, <http://localhost:8080/> にアクセスする。

3, テキストボックスに文字を入れる。

4, Sendボタンを押下。

5, 画面下部に、テキストが送信される。

← → ↻ ⓘ localhost:8080

Customer Information

Name	Address	Telno
<input type="text" value="Alex"/>	<input type="text" value="California"/>	<input type="text" value="00000"/>
<input type="text" value="Dana"/>	<input type="text" value="Florida"/>	<input type="text" value="11111"/>
<input type="text" value="Jamie"/>	<input type="text" value="Iowa"/>	<input type="text" value="22222"/>
<input type="text" value="Morgan"/>	<input type="text" value="Kansas"/>	<input type="text" value="33333"/>
<input type="text" value="Robin"/>	<input type="text" value="Idaho"/>	<input type="text" value="44444"/>
<input type="text" value="Terry"/>	<input type="text" value="Alaska"/>	<input type="text" value="55555"/>
<input type="text" value="Chris"/>	<input type="text" value="Texas"/>	<input type="text" value="66666"/>
<input type="text" value="Pat"/>	<input type="text" value="Vermont"/>	<input type="text" value="77777"/>
<input type="text" value="Tracey"/>	<input type="text" value="Wisconsin"/>	<input type="text" value="88888"/>
<input type="text" value="Ronnie"/>	<input type="text" value="Utah"/>	<input type="text" value="99999"/>



← → ↻ ⓘ localhost:8080/post?members%5B0%5D.name=Alex&members%5B0%5D.address=California&r

<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>

Alex California 00000
Dana Florida 11111
Jamie Iowa 22222
Morgan Kansas 33333
Robin Idaho 44444
Terry Alaska 55555
Chris Texas 66666
Pat Vermont 77777
Tracey Wisconsin 88888
Ronnie Utah 99999

Spring Boot 基本学習

Fin.