6-5.エラーメッセージを表示する

エラーメッセージを表示させる

エラー時に送信ボタンを押したら、エラーメッセージを表示するアプリケーションを作成します。

お問い合わせ

入力してください

お名前

空白は許可されていません

メールアドレス

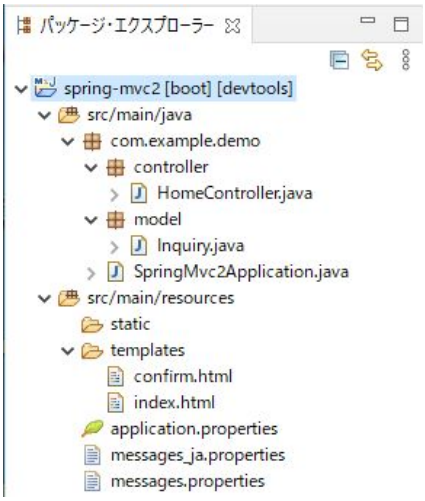
空白は許可されていません

お問い合わせ

空白は許可されていません

登録

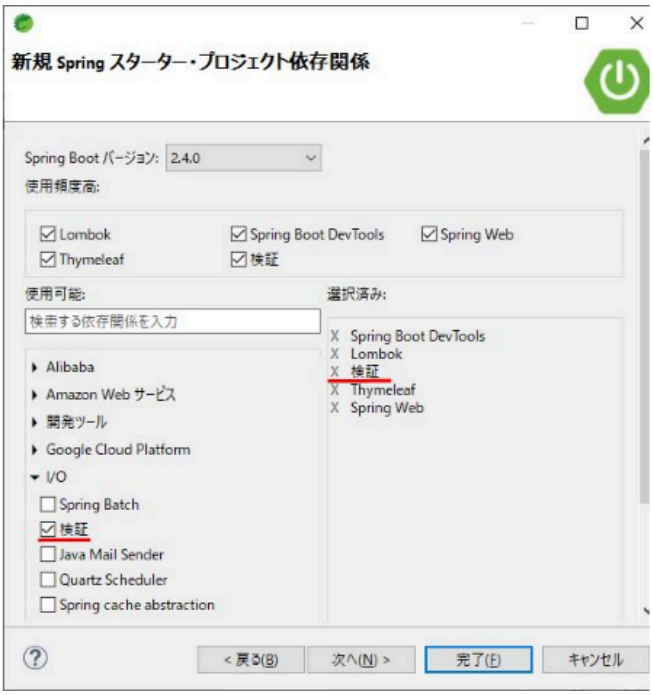
フォルダ構成



プロジェクト依存関係は下記です

- 開発ツール→ **Spring Boot DevTools** と **Lombok**
- テンプレートエンジン→ **Thymeleaf**
- Web→ **Spring Web**
- I/O→ **検証**

?



を選択し、完了を押下する。

モデルの作成

Inquiry.java を以下のように編集します。

```
package com.example.demo.model;

import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class Inquiry {

        // 必須入力、文字列が60文字まで
        @NotBlank
        @Size(max = 60)
        private String name;

        // 必須入力、Email形式であること、文字列が254文字まで
        @NotBlank
        @Email
        @Size(max = 254)
        private String email;

        // 必須入力、文字列が500文字まで
        @NotBlank
        @Size(max = 500)
        private String inquiry;
}
```

@NotBlank や **@Size** は、バリデーション用のアノテーションです。
これらを使うことで、エラーチェックが簡単に実装できます。

代表的なバリデーションを以下に記載します。

アノテーション	内容	使用例
@NotBlank	文字列が、nullか空文字、空白(半角スペース)でないかを検証	@NotBlank String name;
@NotEmpty	文字列が、nullか空文字でないかを検証	@NotEmpty String name;
@NotNull	nullでないかを検証	@NotNull String id;
@Size	文字列の長さが、指定の範囲内かを検証	@Size(min=4, max=100) String name;

アノテーション	内容	使用例
@Email	文字列が、有効なメールアドレスかを検証	@Email String email;
@Min	指定の値以上かを検証	@Min(1) Integer age;
@Max	指定の値以下かを検証	@Max(120) Integer age;
@Pattern	文字列が、正規表現と一致するかを検証	@Pattern(regexp[0-9]+") String phoneNumber;
@AssertFalse	Falseであるかを検証	@AssertFalse Boolean isActive;
@AssertTrue	Trueであるかを検証	@AssertTrue Boolean isActive;
@Future	日付が未来かどうかを検証	@Future Date eventDate;
@Past	日付が過去かどうかを検証	@Past Date eventDate;

コントローラーの作成

HomeController.java を以下のように編集します。

```
package com.example.demo.controller;

import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import com.example.demo.model.Inquiry;

@Controller
public class HomeController {
	@GetMapping("/")
	public String index(@ModelAttribute Inquiry inquiry) {
		return "index";
	}

	@PostMapping("/")
	public String confirm(@Validated @ModelAttribute Inquiry inquiry, BindingResult result) {

		if (result.hasErrors()) {
			// エラーがある場合、index.htmlに戻る
			return "index";
		}
		return "confirm";

	}
}
```

入力チェック

`@Validated` を使うことで、入力値のチェックを行います。
チェックの結果は、`BindingReult` に入るので `result.hasErrors()` でエラーがあるか確認できます。

- **hasErrors メソッド** の戻り値が **true** の場合　・・・　入力エラーが発生している
- **hasErrors メソッド** の戻り値が **false** の場合　・・・　入力エラーが発生している

今回の場合、 **if文 (条件分岐)** で `hasErrors` メソッドの結果が **true** 、つまり、入力エラーがある場合に **index.html** が呼び出され、`hasErrors` メソッドの結果が **false** の場合は入力エラーがないということで、 **confirm.html** が呼び出されます。

入力フォーム の作成

index.html を以下のように編集します。

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Title</title>
</head>
<body>
	<div>
		<h4>お問い合わせ</h4>
		<p>入力してください</p>
		<form th:action="@{/}" th:object="${inquiry}" method="post" novalidate>

			<label for="name">お名前</label><br> <input type="text"
				th:errorclass="is-invalid" th:field="*{name}" />
			<div th:errors="*{name}"></div><br>

			<label for="email">メールアドレス</label><br> <input type="email"
				th:errorclass="is-invalid" th:field="*{email}" />
			<div th:errors="*{email}"></div><br>

			<label for="inquiry">お問い合わせ</label><br>
			<textarea rows="3" th:errorclass="is-invalid" th:field="*{inquiry}"></textarea>
			<div th:errors="*{inquiry}"></div><br>

			<button type="submit">登録</button>
		</form>
	</div>
</body>
</html>
```

エラーメッセージを表示する

- **th:errors 属性**　・・・ エラーメッセージを取得します。
  *{プロパティ名} の形式で、どのプロパティに対するエラーメッセージなのかを指定します。
- **th:errorclass 属性**　・・・エラーがある場合に、class属性にスタイルを追加します。
  エラーのチェックは、 **th:fieldまたはname属性から取得するため、これらと一緒に使う必要があります。**

エラーが発生しない場合の遷移先

confirm.html を以下のように編集します。

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Title</title>
</head>
<body>
	<div th:object="${inquiry}">
		<h4>お問い合わせ</h4>
		<p>以下の内容で登録しました</p>
		<div>
			<label for="name">お名前</label><br> <input type="text"
				th:field="*{name}" disabled />
		</div>

		<div>
			<label for="email">メールアドレス</label><br> <input type="email"
				th:field="*{email}" disabled />
		</div>

		<div>
			<label for="inquiry">お問い合わせ</label><br>
			<textarea rows="3" th:field="*{inquiry}" disabled></textarea>
		</div>
	</div>

</body>
</html>
```

ソースコードの記述はこれで完了です。
最後に実行し、 http://localhost:8080/ にアクセスししてください。
入力エラーが適用されるか試してみましょう！

バリデーションにはいろいろな機能があるのでどれを使うか迷いますが、必須知識なので流れだけでも是非覚えておきましょう

最終更新日時: 2022年 09月 10日(土曜日) 09:34