

---

# これから始める SpringのWebアプリケーション ～ハンズオン編～

2017/11/24

Starlight & Storm

大野 渉

---

# 自己紹介

---

- 大野 渉
- Starlight & Storm所属
- 仕事
  - 講師、技術支援
    - オブジェクト指向
    - Spring等のオープンソースフレームワーク
    - etc



[改訂新版]Spring入門

# ハンズオン概要

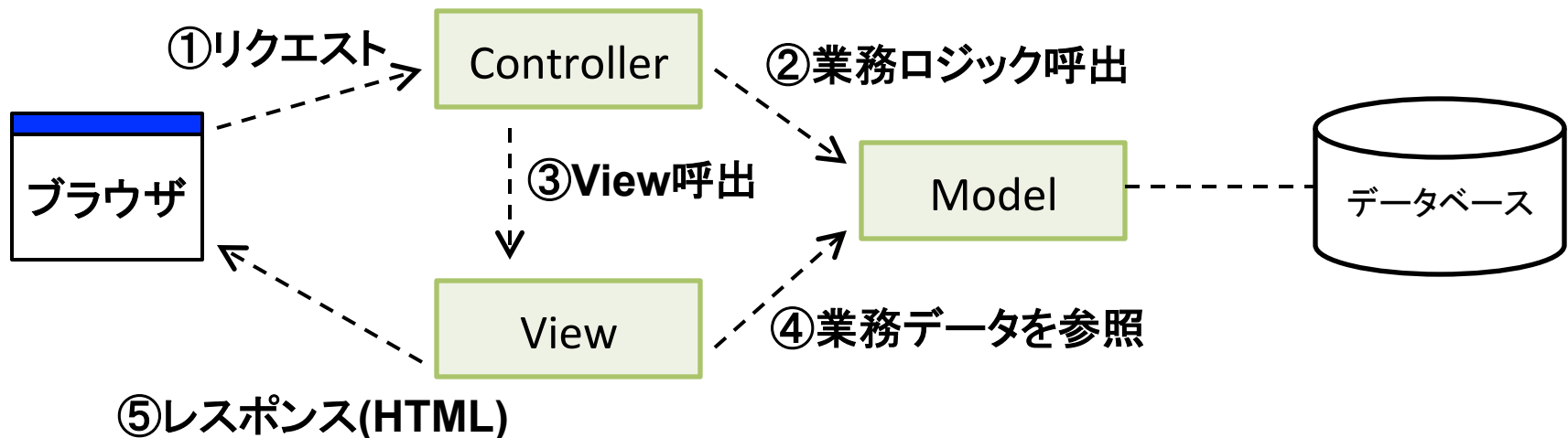
---

- 目的
  - Spring Bootをベースとしたアプリケーションを使ってSpringの開発に触れる
- 使用技術
  - Spring Boot
  - Spring MVC
  - DI/AOP (Option)

**45分で！！**

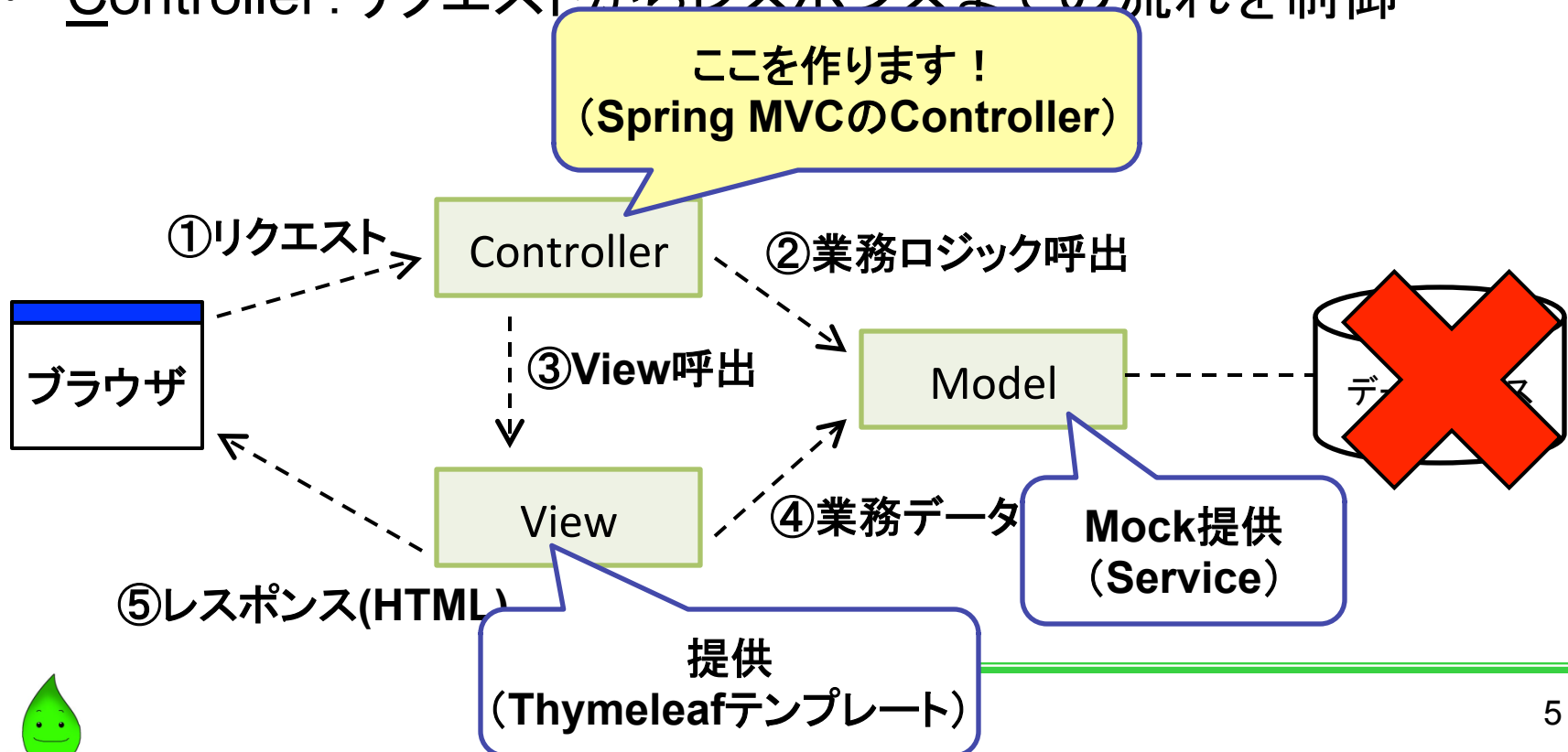
# MVCパターン

- 画面周りのプログラムの効率的な設計の考え方
- Model: 業務的なロジック・データ
- View: 画面(HTMLデータの生成)
- Controller: リクエストからレスポンスまでの流れを制御



# MVCパターン

- 画面周りのプログラムの効率的な設計の考え方
- Model: 業務的なロジック・データ
- View: 画面(HTMLデータの生成)
- Controller: リクエストからレスポンスまでの流れを制御



# ハンズオンの流れ

---

- Exercise 0 - 動作確認
- Exercise 1 - Hello World
- Exercise 2 - ビューにオブジェクトを渡す
- Exercise 3 (Option) - DI/AOPを適用する
- Exercise 4 - 画面から入力値を受け取る

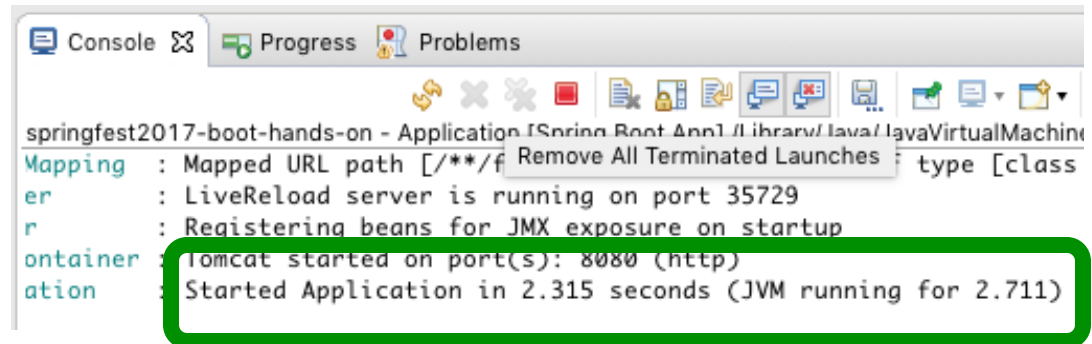
---

# Exercise 0

## 動作確認

# Exercise 0

- 動作確認をおこなう
- 手順
  - 以下のプロジェクトで右クリックし、[Run As] -> [Spring Boot App] を選択する
    - springfest2017-boot-hands-on
- 動作確認
  - コンソールの最後に以下が表示されることを確認する
    - Started Application in xxx seconds (JVM running for xxx)



```
springfest2017-boot-hands-on - Application [Spring Boot App] / I:\hrarv\Java\JavaVirtualMachine
Mapping : Mapped URL path [/**/*] type [class
er      : LiveReload server is running on port 35729
r       : Registering beans for JMX exposure on startup
ontainer : tomcat started on port(s): 8080 (http)
ation   : Started Application in 2.315 seconds (JVM running for 2.711)
```



---

# Exercise 1

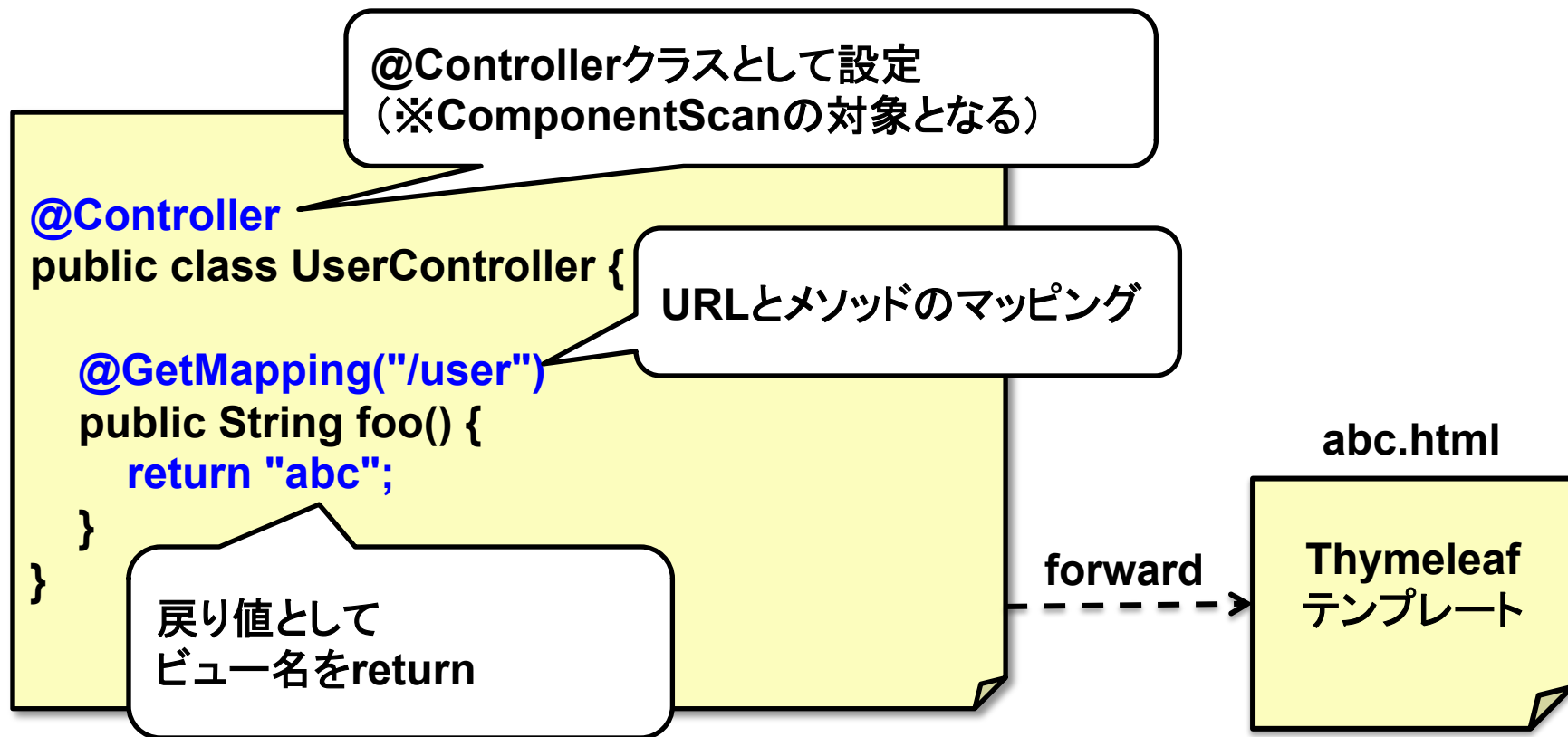
## Hello World

# Exercise 1 概要

---

- 1からControllerを作成して動かしてみる
  - Ex1.1 Controller作成
  - Ex1.2 リクエストパラメータ取得
  - Ex1.3 ビューにオブジェクトを渡す

# ポイント: Controllerの基本



# ポイント: リクエストパラメータの取得

---

- @RequestParamアノテーションで指定する
  - リクエストパラメータ名と、引数の名前を合わせる

```
@GetMapping("/abc")
public String method1(@RequestParam String param1) {
    // リクエストパラメータ「param1」の値が、引数param1に入る
    // 例) /abc?param1=valと指定すると、引数param1に"val"が入る
    . . .
}
```

# ポイント: Modelオブジェクト

- Controllerからビューに渡すオブジェクトを格納するオブジェクト
  - Modelオブジェクトに格納したオブジェクトは、自動的にrequestスコープまたはsessionスコープに格納される

```
@GetMapping("/abc")  
public String method1(Model model) {  
    model.addAttribute("message", "こんにちは！");  
}
```

オブジェクトの格納

```
...  
<span th:text="${message}"></span>  
...
```

オブジェクトの取得  
(Thymeleafの例)

# Ex1.1 Controller作成

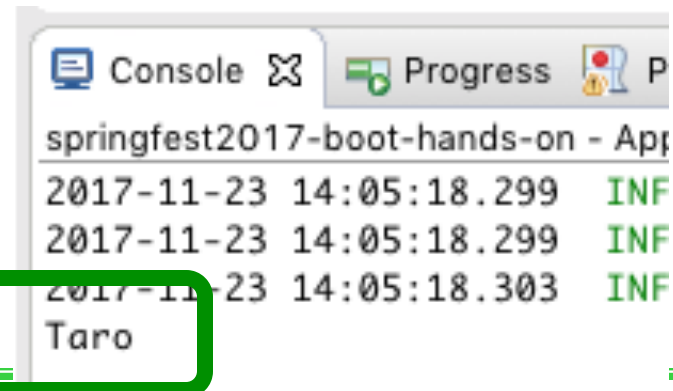
---

- 手順
  - Controllerクラスを新規作成する
    - 作成するクラス
      - ~.ex1. HelloController
    - @Controllerを指定
  - 「String hello()」メソッドを作成する
  - @GetMappingで「/ex1/hello」を指定する
  - 戻り値(遷移先のビュー名)として"hello"を返す
    - 遷移先は「/src/main/resources/templates/hello.html」となる
- 動作確認
  - ブラウザで以下のURLに接続する
    - <http://localhost:8080/ex1/hello>
  - 画面が表示されることを確認する



# Ex1.2 リクエストパラメータ取得

- 手順
  - helloメソッドの引数にリクエストパラメータを指定する
    - @RequestParam String name
  - nameをSystem.out.printlnで表示する
- 動作確認
  - ブラウザで以下のURLに接続する
    - <http://localhost:8080/ex1/hello?name=Taro>
  - コンソール上にTaroが表示されることを確認する



# Ex1.3 ビューにオブジェクトを渡す

- 手順
  - helloメソッドの第2引数にModelオブジェクトを指定する
    - Model model
  - Modelオブジェクトにメッセージを追加する(addAttribute)
    - attribute名(第1引数) : "hello"
    - 追加するオブジェクト(第2引数) : "こんにちは、" + name + "さん！"
- 動作確認
  - ブラウザで以下のURLに接続する
    - <http://localhost:8080/ex1/hello?name=Taro>
  - 画面に「こんにちは、Taroさん！」が表示されることを確認する





# (参考)hello.html

---

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8"/>
<title>Hello World</title>
</head>
<body>
Helloをここに表示します:
<span th:text="${hello}"></span>
</body>
</html>
```

# Exercise 1 解答

---

```
@Controller
public class HelloWorldController {

    @GetMapping("/ex1/hello")
    public String showHello(@RequestParam String name, Model model ) {
        System.out.println(name);
        model.addAttribute("hello", "こんにちは、" + name + "さん！！");

        return "hello";
    }
}
```

---

## Exercise 2

# ビューにオブジェクトを渡す

# Exercise 2 概要

---

- ビューに様々なオブジェクトを渡す
  - Ex2.1 Personオブジェクトを渡す
  - Ex2.2 URIテンプレートでURLを指定する
  - Ex2.3 PersonのListを渡す

# ポイント: URIテンプレート

---

- URLの指定をテンプレート化する方法
  - 一部が動的に変わるURLを指定
    - 動的に変わる部分を{名前}の形式で指定
    - 動的に変わる部分の値を容易に取得可能
  - 指定例) 「http://foo.bar/{value}」
    - 「http://foo.bar/123」や「http://foo.bar/456」等でアクセス可能
    - 「123」や「456」を「value」という名前で取得

# ポイント: URIテンプレートと値の取得

---

- {名前}の部分に該当するURL中の値を、引数として取得
  - 引数に@PathVariableアノテーションを指定
  - 引数名とテンプレート内の変数名を合わせる

```
@GetMapping("/foo/{value}")
public String method1(@PathVariable String value) {
    // URLに「/foo/abc123」を指定した場合、valueには「abc123」が入る
    . . .
}
```

# Ex2.1 Personオブジェクトを渡す

- 手順
  - 以下のクラスを開き、@Controllerを指定する
    - ~.ex2.PersonViewController
  - showPersonメソッドに以下を実装する
    - @GetMappingで「/ex2/person」を指定する
    - 戻り値(遷移先のビュー名)として"showPerson"を返す
      - 遷移先は「/src/main/resources/templates/showPerson.html」となる
    - 引数にModelを指定する
      - Model model
    - ModelオブジェクトにPersonオブジェクトを追加する(addAttribute)
      - attribute名(第1引数) : "p"
      - 追加するオブジェクト(第2引数) : person変数
- 動作確認
  - ブラウザで以下のURLに接続
    - <http://localhost:8080/ex2/person>
  - 画面が表示されることを確認



# (参考) showPerson.html

---

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8"/>
<title>Person表示</title>
</head>
<body>
<div th:if="${p != null}">
  <span th:text="${p.id}">仮番号</span>番の
  <span th:text="${p.name}">仮名前</span>です。
  <span th:text="${p.address}">仮住所</span>に住んでいます。
</div>
</body>
</html>
```



## Ex2.2 URIテンプレートでURLを指定する

- 手順
  - showPersonメソッドのURLマッピングを以下に変更する
    - /person/{id}
  - 第2引数にURLで指定した「id」を指定する
    - @PathVariable long id
  - Personコンストラクタ呼出の第1引数に、idを指定する
    - new Person(id, "太郎", "東京都");
- 動作確認
  - ブラウザで以下のURLに接続
    - <http://localhost:8080/ex2/person/300>
  - 画面が表示されることを確認



# Ex2.3 PersonのListを渡す

- 手順
  - showPersonListメソッドに以下を実装する
    - @GetMappingで「/ex2/person」を指定する
    - 戻り値(遷移先のビュー名)として"showPersonList"を返す
      - 遷移先は「/src/main/resources/templates/showPersonList.html」となる
    - ModelオブジェクトにPersonオブジェクトを追加する(addAttribute)
      - attribute名(第1引数) : "pList"
      - 追加するオブジェクト(第2引数) : list変数
- 動作確認
  - ブラウザで以下のURLに接続
    - <http://localhost:8080/ex2/person>
  - 画面が表示されることを確認



# (参考) showPersonList.html

---

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8"/>
<title>Person一覧</title>
</head>
<body>
<ul th:if="${pList != null}">
  <li th:each="p : ${pList}">
    <span th:text="${p.name}">仮の名前</span>です。
    <span th:text="${p.address}">仮の住所</span>に住んでいます。
  </li>
</ul>
</body>
</html>
```

# Exercise 2 解答

---

```
@Controller
@RequestMapping("/ex2")
public class PersonViewController {

    @GetMapping("/person/{id}")
    public String showPerson(@PathVariable long id, Model model) {
        Person person = new Person(id, "太郎", "東京都");
        model.addAttribute("p", person);

        return "showPerson";
    }
}
```

(次ページへ続く)

# Exercise 2 解答

---

(次ページから続き)

```
@GetMapping("/person")
public String showPersonList(Model model) {
    List<Person> list = new LinkedList<>();

    list.add(new Person(100, "太郎", "東京都"));
    list.add(new Person(200, "次郎", "神奈川県"));
    list.add(new Person(300, "三郎", "埼玉県"));

    model.addAttribute("pList", list);

    return "showPersonList";
}
```

---

# Exercise 3 (Option)

## DI/AOPを適用する

# Exercise 3 概要

---

- DIとAOPを適用する
  - Ex3.1 DIする(1)
  - Ex3.2 DIする(2)
  - Ex3.3 AOPを適用する

# Ex3.1 DIする(1)

---

- 手順
  - 以下のクラスを開く
    - ~.ex3.PersonViewUsingServiceController
  - インスタンス変数としてPersonServiceを定義する
    - `private PersonService personService;`
  - PersonServiceを受け取るコンストラクタを定義する
    - 右クリック -> [Source] -> [Generate Constructor using Fields]
  - showPersonメソッドのperson変数に、`personService.findById(id)`の結果を代入する
    - `Person person = personService.findById(id);`
- 動作確認
  - ブラウザで以下のURLに接続
    - `http://localhost:8080/ex3/person/300`
  - 画面が表示されることを確認





## Ex3.2 DIする(2)

- 手順
  - showPersonListメソッドのlist変数に、personService.findAll()の結果を代入する
    - `Person person = personService.findAll();`
- 動作確認
  - ブラウザで以下のURLに接続
    - `http://localhost:8080/ex3/person`
  - 画面が表示されることを確認



# Ex3.3 AOPを適用する

---

- 手順
  - 以下のクラスを開き、@Aspect, @Componentを設定する  
(Aspectクラスは、DIコンテナにBeanとして登録する必要がある)
    - ~.aspect.SysoutAspect
  - beforeメソッドに@Beforeアノテーションを指定する
    - @Before("execution(\* com.example..\*Service.\*(..))")
  - afterメソッドに@Afterアノテーションを指定する
    - @After("execution(\* com.example..\*Service.\*(..))")
- 動作確認
  - ブラウザで以下のURLに接続
    - <http://localhost:8080/ex3/person/300>
    - <http://localhost:8080/ex3/person>
  - コンソールに、「メソッド実行後に表示」「メソッド実行前に表示」が表示されることを確認する

# Exercise 3 解答 (Controller)

---

```
@Controller
@RequestMapping("/ex3")
public class PersonViewUsingServiceController {

    private PersonService personService;

    public PersonViewUsingServiceController(PersonService personService) {
        this.personService = personService;
    }

    @GetMapping("/person/{id}")
    public String showPerson(@PathVariable long id, Model model) {
        Person person = personService.findById(id);
        model.addAttribute("p", person);

        return "showPerson";
    }

    @GetMapping("/person")
    public String showPersonList(Model model) {
        List<Person> list = personService.findAll();
        model.addAttribute("pList", list);

        return "showPersonList";
    }
}
```

# Exercise 3 解答 (Aspect)

---

```
@Aspect
@Component
public class SysoutAspect {

    @Before("execution(* com.example..*Service.*(..))")
    public void before() {
        System.out.println("メソッド実行前に表示");
    }

    @After("execution(* com.example..*Service.*(..))")
    public void after() {
        System.out.println("メソッド実行後に表示");
    }
}
```

---

## Exercise 4

– 画面から入力値を受け取る –

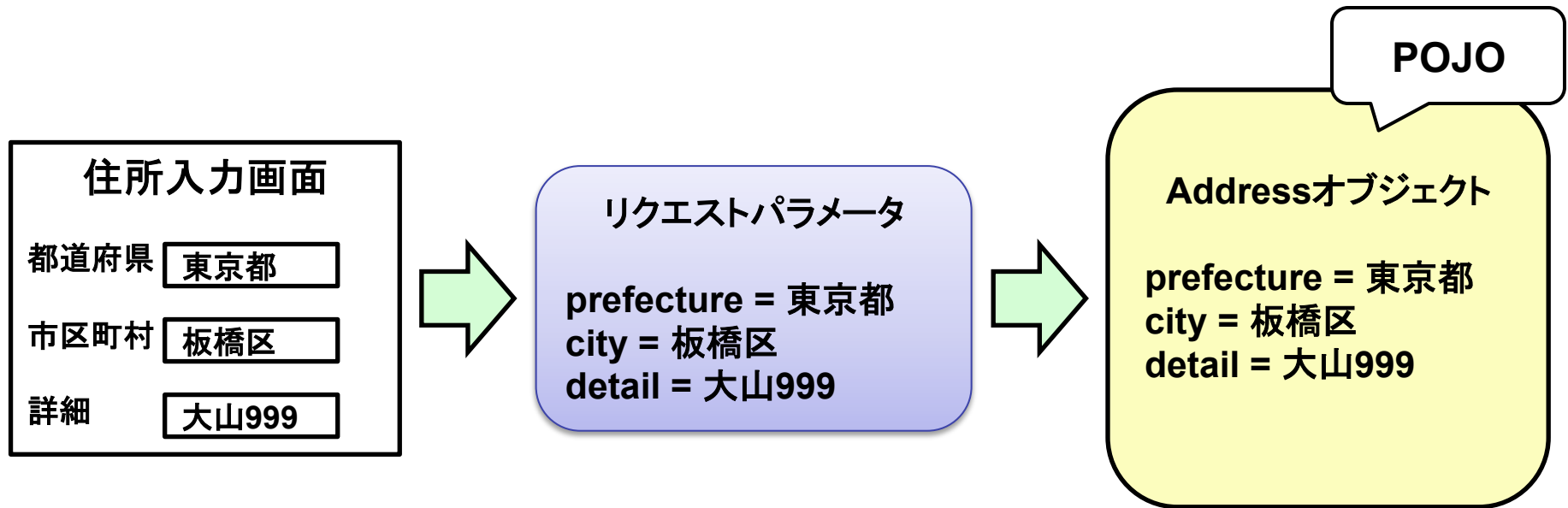
# Exercise4 概要

---

- Controllerで入力値を受け取る
  - Ex4.1 入力画面を表示する
  - Ex4.2 入力値を受け取る

# ポイント: 画面の入力値の受け取り

- @RequestParam変数
  - 項目が多いと引数が膨大になる
- ModelAttributeオブジェクト
  - Controllerメソッドの引数として定義されたオブジェクトが自動で生成され、入力値も自動で設定される



# ポイント:ModelAttributeオブジェクトの指定

- 引数に定義する

```
@PostMapping("/updateAddress")
public String updateAddress(Address address) {
    // リクエストパラメータの値をaddressから取得できる
}
```

```
class Address {
    private String prefecture;
    private String city;
    private String detail;
    // 以下、Getter/Setter省略
}
```

リクエストパラメータ名と一致させることで、自動的にリクエストパラメータ値が設定される

リクエストパラメータ

prefecture = 東京都  
city = 板橋区  
detail = 大山999



## ポイント:ModelAttributeオブジェクトとModelオブジェクト

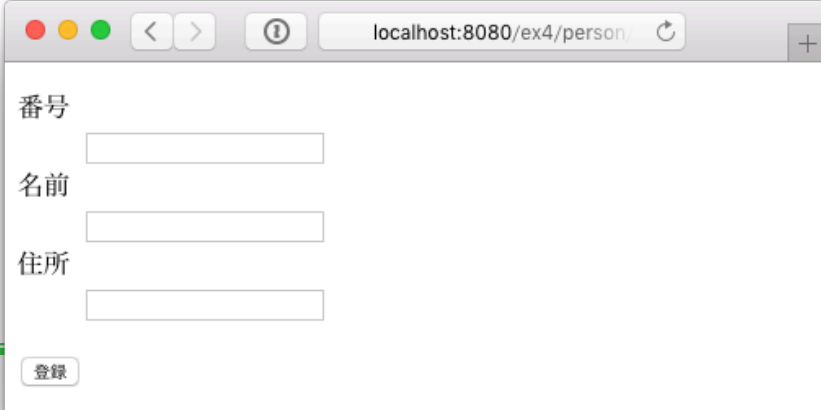
---

- ModelAttributeオブジェクトは自動的にModelオブジェクトに格納される
  - ModelのaddAttributeメソッドを実行した場合と同じ処理が実行される
  - クラス名の先頭を小文字にした文字列が、キーとして設定される
  - @ModelAttributeで明示的にキーを設定することも可能

```
@PostMapping("/updateAddress")
public String updateAddress(
    @ModelAttribute("userAddress") Address address) {
    ...
}
```

# Ex4.1 入力画面を表示する

- 手順
  - 以下のクラスを開き、@Controllerを指定する
    - ~.ex4.PersonInputController
  - showFormメソッドに以下を実装する
    - @GetMappingで「/ex4/person/input」を指定する
    - 戻り値(遷移先のビュー名)として"inputPerson"を返す
      - 遷移先は「/src/main/resources/templates/inputPerson.html」となる
- 動作確認
  - ブラウザで以下のURLに接続
    - <http://localhost:8080/ex4/person/input>
  - 画面が表示されることを確認



番号

名前

住所

登録

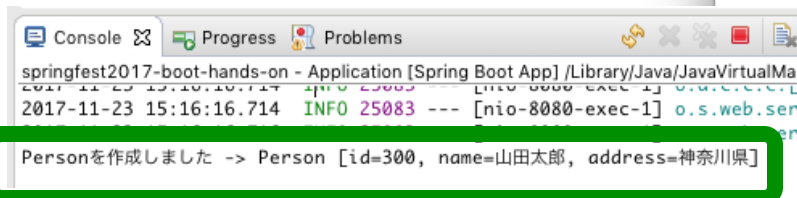
# (参考)inputPerson.html

---

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8"/>
<title>Person入力</title>
</head>
<body>
<form th:action="@{/ex4/person/input}" method="post">
  <dl>
    <dt>番号</dt>
    <dd><input type="text" name="id" /></dd>
    <dt>名前</dt>
    <dd><input type="text" name="name" /></dd>
    <dt>住所</dt>
    <dd><input type="text" name="address" /></dd>
  </dl>
  <button>登録</button>
</form>
</body>
</html>
```

# Ex4.2 入力値を受け取る

- 手順
  - onSubmitメソッドに以下を実装する
    - @PostMappingで「/ex4/person/input」を指定する
    - 引数にPersonを指定する
      - @ModelAttribute("p") Person person
      - (Personオブジェクトをattribute名「p」でModelに格納している)
    - PersonServiceのcreateメソッドを実行する
      - personService.create(person);
    - 戻り値(遷移先のビュー名)として"showPerson"を返す
      - 遷移先は「/src/main/resources/templates/showPerson.html」となる
- 動作確認
  - ブラウザで以下のURLに接続
    - <http://localhost:8080/ex4/person/input>
  - 番号、名前、住所を入力して登録ボタンを押下
  - 入力した情報が表示されることを確認
  - コンソールにメッセージが表示されることを確認



# Exercise 4 解答

---

```
@Controller
@RequestMapping("/ex4")
public class PersonInputController {

    private PersonService personService;

    public PersonInputController(PersonService personService) {
        this.personService = personService;
    }

    @GetMapping("/person/input")
    public String showForm() {
        return "inputPerson";
    }

    @PostMapping("/person/input")
    public String onSubmit(@ModelAttribute("p") Person person) {
        personService.create(person);
        return "showPerson";
    }
}
```

---

ご清聴ありがとうございました

# ライセンスについて

---

- ① JSUGマスコットアイコン(本スライド左下)が残されている場合に限り、本作品(またそれを元にした派生作品)の複製・頒布・表示・上演を認めます。
- ② 非商用目的に限り、本作品(またそれを元にした派生作品)の複製・頒布・表示・上演を認めます。
- ③ 本作品のライセンスを遵守する限り、派生作品を頒布することを許可します。