フレームワーク基礎実習

06. リクエスト処理

株式会社ジードライブ

今回学ぶこと

- 外部入力値の取得に関する用語と概念
- GETパラメーターとPOSTパラメーター
- URIテンプレート変数
- フォワードとリダイレクト

Webシステムの機能

- 多くのWebシステムは以下のような機能を持つ

① ユーザが何らかの情報を**入力**する機能
例：ログイン情報など

② 入力データを**取得して処理**する機能
例：入力データのチェックなど

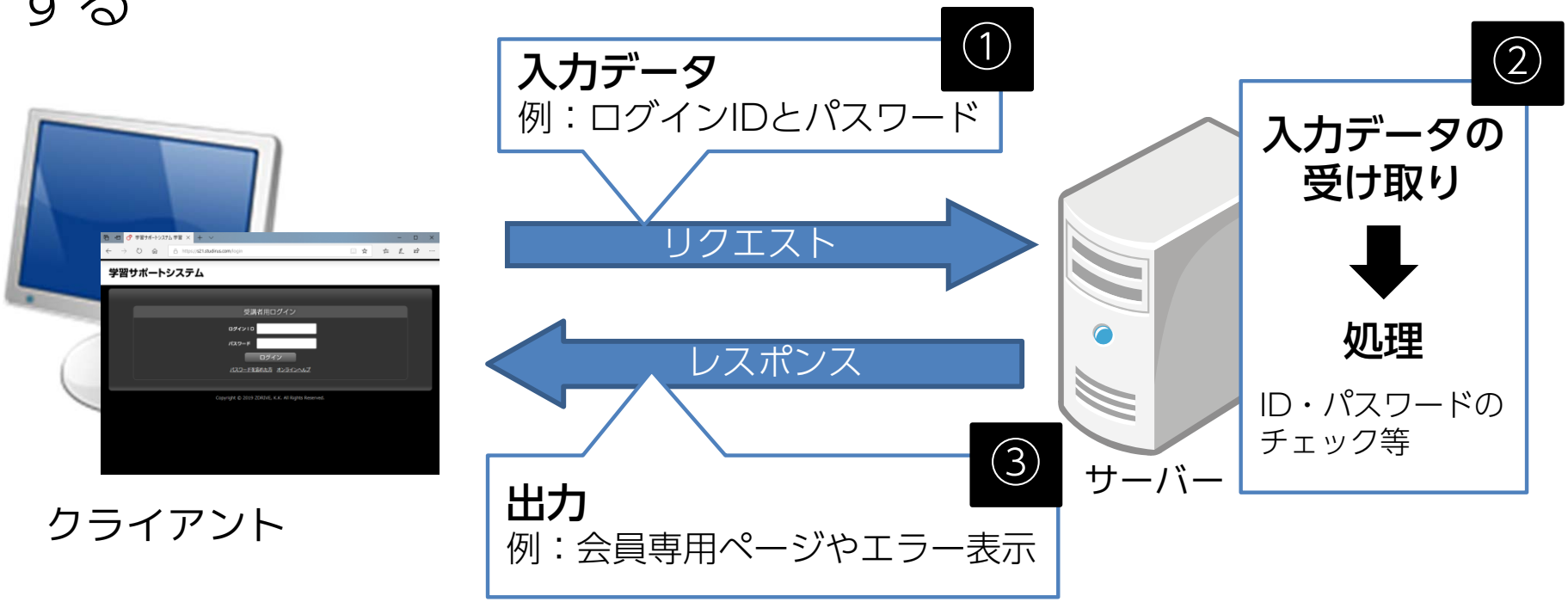
③ 処理後に何らかの**出力**を行う機能
例：ログイン成功の表示やエラーの表示など

Webシステムの入力～出力

① クライアントが **HTTPリクエスト**を通じ、サーバーへ入力データを送る

② サーバー側で入力データを**取得し、処理する**

③ クライアントへの**レスポンス**として、何かしらの**出力**をする

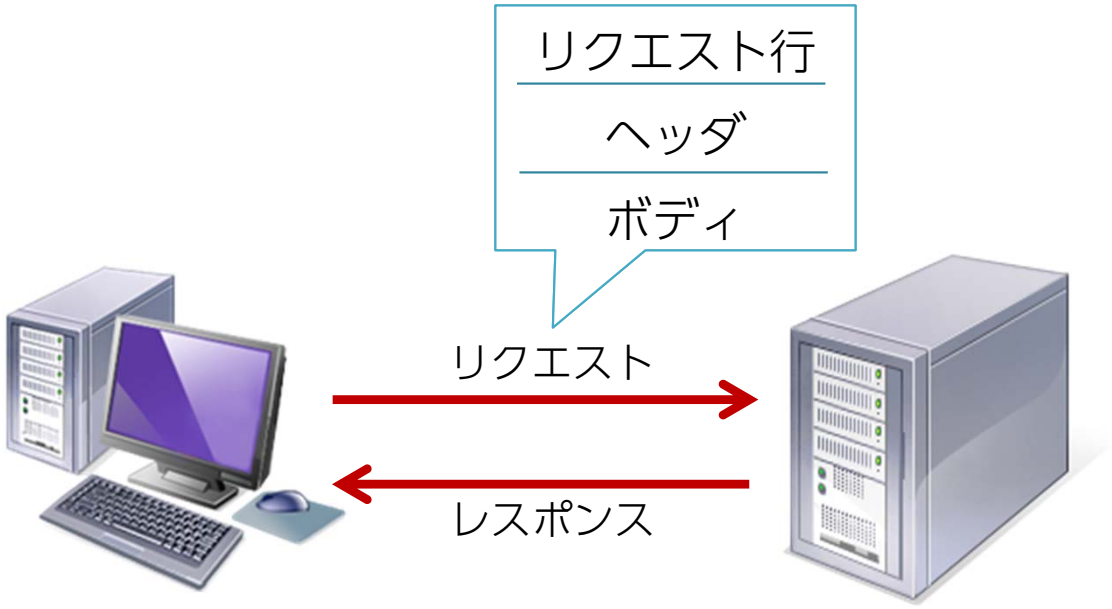


リクエストの確認

①リクエスト行： **メソッド(リクエストの種類：GETまたはPOST)**、
要求するファイルのURL、HTTPのバージョン

②ヘッダ： クライアントからサーバに渡す情報
ブラウザ、COOKIE、言語など

③ボディ： メソッドがPOSTの場合、フォームから送信されるデータ
(POSTパラメータ)



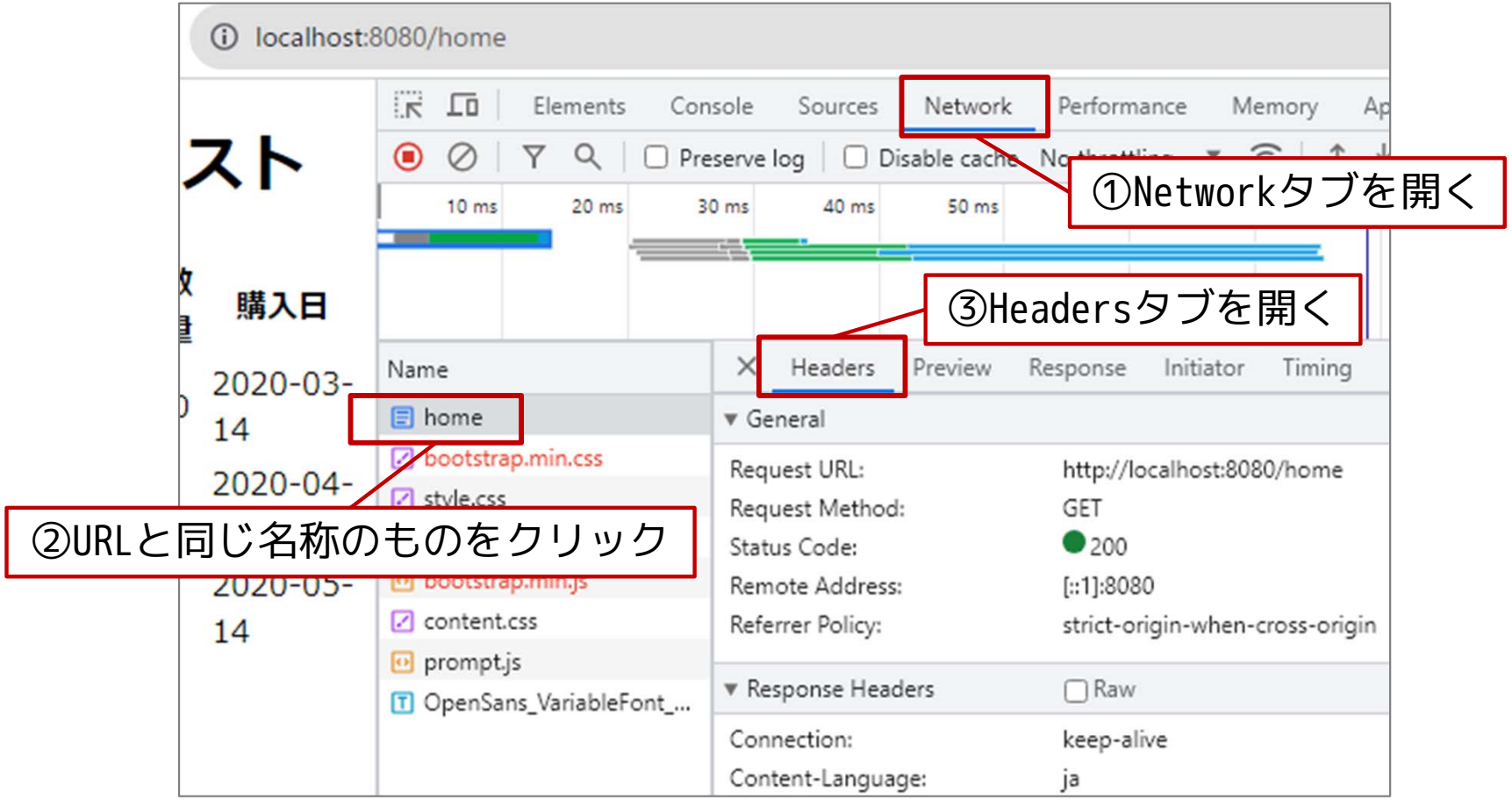
HTTPリクエストの例

```
GET /index.html HTTP/1.1
Accept:text/html,application/xhtml+xml,*/*
Accept-Encoding:gzip, deflate
Accept-Language:ja
Connection:keep-alive
Cookie:_ga=GA1.2.43392.15016582
Host:www.zdrv.com
User-Agent:Mozilla/5.0 (Windows NT 10.0;
Win64; x64)
```

リクエスト行
ヘッダ
ボディ

リクエストの確認

- Chromeの開発者ツールでは、Networkタブを開くことで、リクエストの情報を確認することができる



リクエストの種類

- リクエストには幾つかの種類（リクエストメソッド）が存在する
- おもなリクエストメソッドとしては、GETリクエストとPOSTリクエストが存在する
  - **@GetMapping**, **@PostMapping**アノテーションを使うことで、それぞれのリクエストメソッドに対応することができる

おもなリクエストメソッド

メソッド	説明
GET	通常の（直接のURL入力や検索、リンクからの）ページアクセス時に送信されるリクエストメソッド ⇒ @GetMappingを付与したメソッドで対応
POST	フォームの送信ボタンが押された際に発生するリクエストメソッド ⇒ @PostMappingを付与したメソッドで対応

リクエストパラメーター

- クライアントはリクエストに併せて、パラメーター送信することができる
  - このパラメーターをリクエストパラメーターと呼ぶ
  - GETリクエストに付与されるものをGETパラメーター、POSTリクエストに付与されるものをPOSTパラメーターと呼ぶ
- コントローラーメソッドでは、リクエストパラメーターを取得するための方法が用意されている

練習問題

- 練習06-1

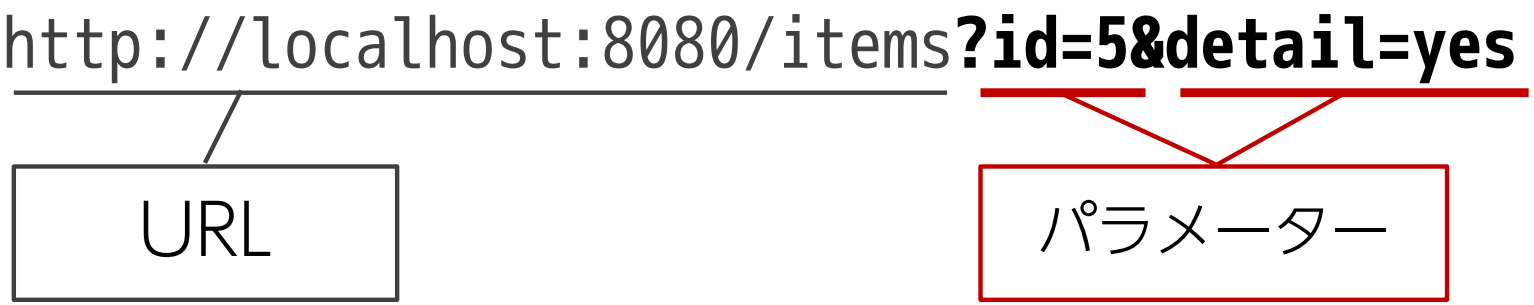
GETパラメーター

GETパラメーター

- URL末尾に付与されるパラメーターで、?以降の文字列がGETパラメーターに相当する
  - クエリストリング(クエリ文字列)とも呼ばれる

例：YouTubeのURL
末尾のGETパラメーターに応じて、表示される動画が決定する

GETパラメーターの構成



- URLの後に ? を付け、パラメーター名とそれに対応する値を = でつなげる
  - 複数のパラメーターがある場合は & で接続する
  - = の前後に空白を入れてはいけない
  - = の後には値がなくてもよい
    （この場合、パラメーターの値は空文字となる）

GETパラメーターの送信方法(1)

方法1:
aタグ内のhref属性値のURL末尾にパラメーターを記述する

```
<li><a th:href="@{/itemDetail?id=1}">りんご</a></li>
<li><a th:href="@{/itemDetail?id=2}">みかん</a></li>
<li><a th:href="@{/itemDetail?id=3}">ぶどう</a></li>
```

複数の値を同時に送信する場合は&で区切る

```
<li><a th:href="@{/room?type=2C&area=18}">2LDK(東伏見)</a></li>
<li><a th:href="@{/room?type=2B&area=23}">2DK(武蔵境)</a></li>
<li><a th:href="@{/room?type=1C&area=18}">1DK(東伏見)</a></li>
```

GETパラメーターの送信方法(2)

方法2：
formタグのmethod属性値をgetに設定する
(method属性の初期値はgetなので、明示的に記述しなくてもよい)

```
<form th:action="@{/itemDetail}" method="get">
    <select name="id">
        <option value="0">りんご</option>
        <option value="1">みかん</option>
        <option value="2">ぶどう</option>
    </select>
    <input type="submit">
</form>
```

th:action属性、name属性、value属性がURLとパラメーターを形成する

送信

```
送信時のURL: localhost:8080/itemDetail?id=0
```

GETパラメーターの取得方法

- コントローラーメソッドの引数に、@RequestParamアノテーションを付与することで、パラメーターを受け取り、メソッド内で利用することができる
  - パラメーターは文字列として送信されるが、コントローラーメソッドで指定した引数の型に変換が行われる

```
URLの例: localhost:8080/itemDetail?id=0
```

送信

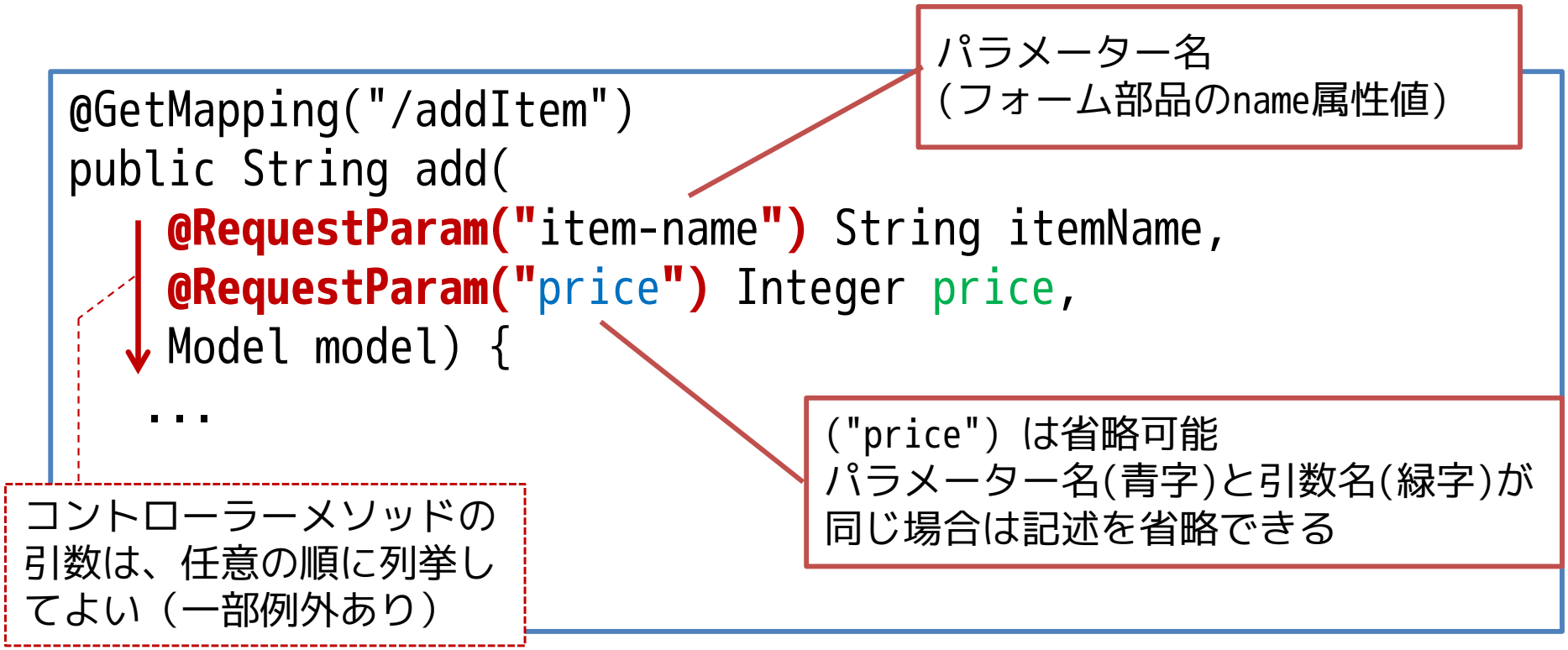
"0"が整数0に変換される

```
@GetMapping("/itemDetail")
public String showDetail(@RequestParam(name="id") int idNum) {
    ...
}
```

name= は省略可

GETパラメーターの取得方法

- 複数のパラメーターを取得する場合は、@RequestParamアノテーションを付与した引数を列挙する



パラメーターの設定

- リクエストパラメーターは、nullの場合エラーになってしまう
  - nullを許可する、または初期値を設定することで対応できる

例：nullを許可する

```
@RequestParam(name = "id", required = false) Integer id
```

例：初期値を設定する

```
@RequestParam(name = "id", defaultValue = "1") Integer id
```

この例の場合、name="id" は省略可
ただし、nameだけ省略することはできない

GETパラメーターの利用場面

- ユーザが取得したい情報の絞り込みのために使われることが多い
  - 検索のキーワード
  - ニュースのカテゴリ
  - 表示したいページ番号
  - 表示したい商品のID番号
- ユーザはGETパラメーターを含めて、URLを共有したり、ブラウザにブックマークしたりすることが可能
- パラメーターがURLに表示されるため、個人情報の扱いには向いていない

練習問題

- 練習06-2

POSTパラメーター

POSTパラメーターの利用場面

- GETパラメーターと異なり、 POSTパラメーターはURLの末尾に表示されない
- ユーザが個人情報を送信する場面で使われることが多い
  - ログイン用ユーザIDとパスワード
  - 問い合わせフォームの内容
  - ユーザ登録の内容

POSTパラメータの送信方法

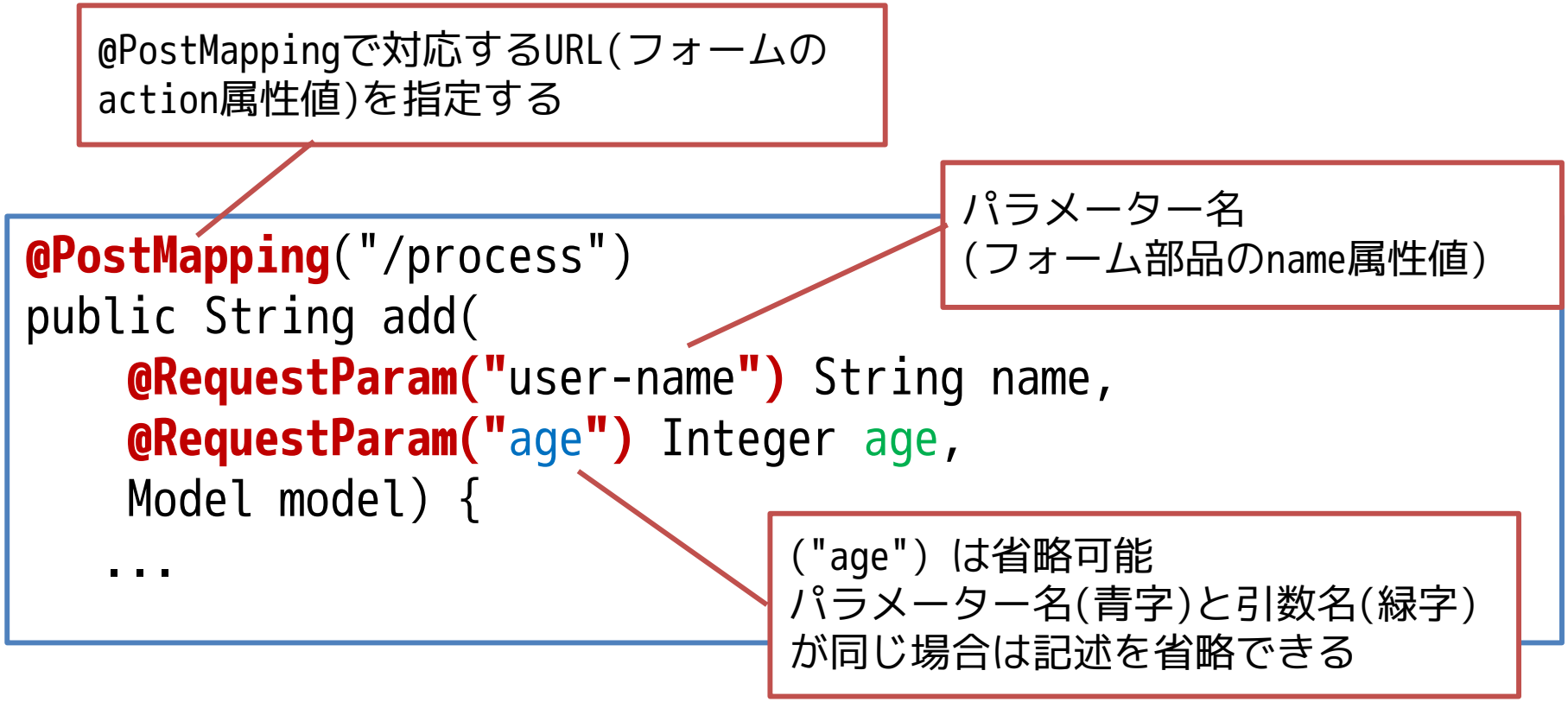
- POSTパラメーターは、method属性にpostを設定したフォームから送信される
  - 送信されたパラメーターをコントローラーメソッドで利用するためには、フォーム部品にname属性の設定が必要

```
<form th:action="@{/process}" method="post">
   氏名: <input type="text" name="user-name">
   年齢: <input type="number" name="age">
   <input type="submit">
</form>
```

送信ボタンは<form>～</form>内に配置する

Postパラメーターの取得方法

- Postパラメーターを取得する方法は、Getパラメーターを取得する場合と同じく、@RequestParamアノテーションを付与した引数を列挙する



現在のURLに送信するフォーム

- formタグ内のaction属性の値を空文字にすると、現在と同じURLに送信するフォームとなる

```
@GetMapping("/register")
public String showRegisterForm() {
    return "registerForm"; //フォームの表示
}

@PostMapping("/register") //GetMappingと同じURLに対応
public String processRegisterForm(@RequestParam ...) {
    ...
}
```

```
<form action="" method="post">
    ...
    <input type="submit">
</form>
```

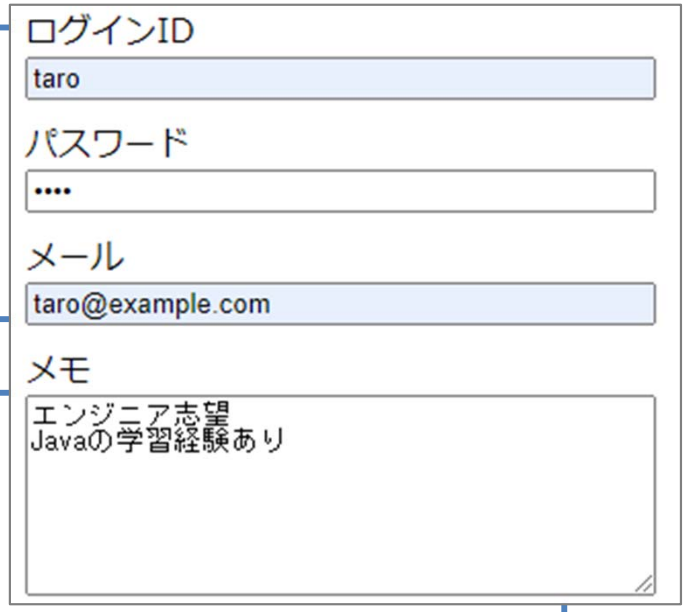
registerForm.html

送信ボタンが押されると現在と同じURLに対応する@PostMappingのメソッドが実行される

テキスト入力欄からのデータ取得

- type属性がtext以外の部品やtextareaからのデータも@RequestParamで取得することができる

```
ログインID <input type="text" name="id">
パスワード <input type="password" name="pass">
メール <input type="email" name="mail">
メモ <textarea name="memo"></textarea>
```

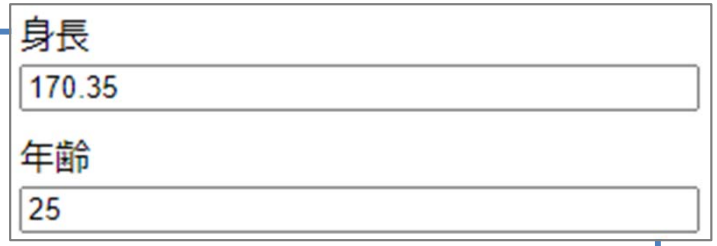



```
@PostMapping("/register")
public String registerProcess(
    @RequestParam String id,
    @RequestParam String pass,
    @RequestParam String mail,
    @RequestParam String memo) {
  ...
```

数値の取得

- <input type="number"> からのデータ取得は、String型だけでなく、int型やdouble型等の引数でもよい

```
身長 <input type="number" name="height"
            step="0.01">
年齢 <input type="number" name="age"
            step="1">
```




```
@PostMapping("/register")
public String registerProcess(
    @RequestParam double height,
    @RequestParam int age) {
  ...
```

数値の取得

- 数値が未入力のまま送信されるとエラーが発生するため、defaultValueやrequiredの設定をする

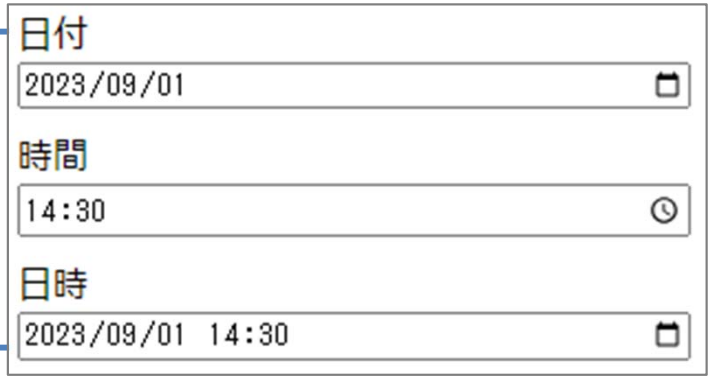
```
@PostMapping("/register")
public String registerProcess(
    @RequestParam(defaultValue = "170.5") double height,
    @RequestParam(required = false) Integer age) {
  ...
```

未入力の場合、ageはnullになるが、int型(プリミティブ型)ではnullを許容できないため、Integer型(ラッパークラスの型)を指定する

日時の取得

- <input type="date"> などの日時を取得する場合は、LocalDate型やLocalTime型等の引数を用意する

```
日付 <input type="date" name="date">
時間 <input type="time" name="time">
日時 <input type="datetime-local"
           name="datetime">
```



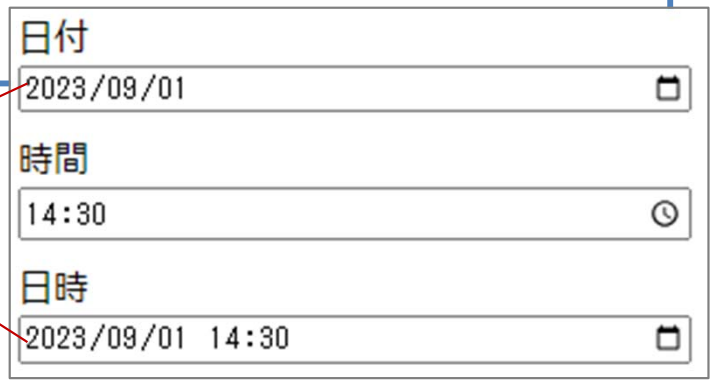

```
@PostMapping("/register")
public String registerProcess(
    @RequestParam(required = false) LocalDate date,
    @RequestParam(required = false) LocalTime time,
    @RequestParam(required = false) LocalDateTime datetime) {
  ...
```

日時の取得

- 日時をjava.util.Date型で取得する場合は、@DateTimeFormatでパターンを指定する

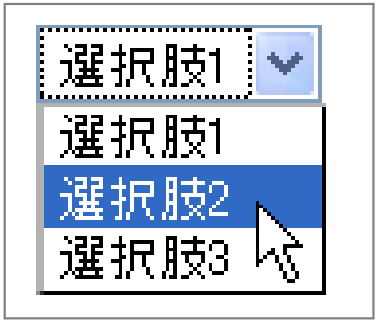
```
@PostMapping("/register")
public String registerProcess(
    @RequestParam(required=false)
        @DateTimeFormat(pattern="y-MM-dd") Date date,
    @RequestParam(required=false)
        @DateTimeFormat(pattern="HH:mm") Date time,
    @RequestParam(required=false)
        @DateTimeFormat(pattern="y-MM-dd'T'HH:mm") Date datetime) {
  ...
```

日付の表記は、スラッシュ区切りになっているが、実際に送信されるデータはハイフン区切りになる



選択式の入力部品(1)

```
<select name="choice">
  <option value="1">選択肢1</option>
  <option value="2">選択肢2</option>
  <option value="3">選択肢3</option>
</select>
```

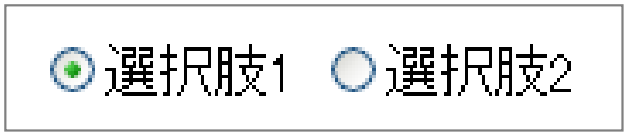



```
@PostMapping("/register")
public String registerProcess(
    @RequestParam int choice) {
  ...
```

選択された項目のvalue属性の値が入る
value属性を省略した場合はoption要素内の文字列が入る

選択式の入力部品(2)

```
<input type="radio" name="choice" value="1">選択肢1
<input type="radio" name="choice" value="2">選択肢2
```

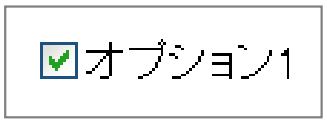



```
@PostMapping("/register")
public String registerProcess(
    @RequestParam(defaultValue = "1") int choice) {
  ...
```

選択された項目のvalue属性の値が入る
value属性を省略した場合は「on」という文字列

チェックボックス(1)

```
<input type="checkbox" name="choice">オプション1
```




```
@PostMapping("/register")
public String registerProcess(
    @RequestParam(required = false) String choice) {
  ...
```

1. チェック(✔)が入っている場合のみ、value値が送信される
   ⇒ チェックが入っていない場合、null が入る
2. value属性を省略した場合は "**on**" が入る

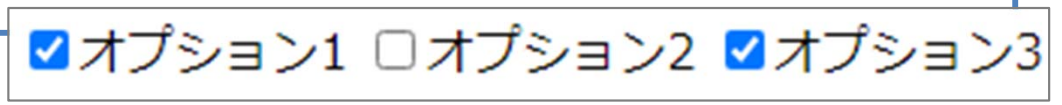
```
@RequestParam(required = false) boolean choice
```

boolean型にすると、未チェックの場合はfalseが、
チェックしている場合はtrueが入る

チェックボックス(2)

- チェックボックスはラジオボタンと違い複数選択が可能

```
<input type="checkbox" name="choice" value="1">オプション1
<input type="checkbox" name="choice" value="2">オプション2
<input type="checkbox" name="choice" value="3">オプション3
```



List型または配列型にして取得する

```
@PostMapping("/register")
public String registerProcess(
    @RequestParam(required = false) List<Integer> choice) {
  if(choice != null) {
    for(int c : choice) { System.out.println(c); }
  }
  ...
```

練習問題

- 練習06-3

URIテンプレート変数

URIテンプレート変数

- URIテンプレート変数とは、
  http://www.example.com/showItem/5
  のような形式でパラメーターを指定できる仕組み
  - Getパラメーターのように扱うことができる
- @PathVariableアノテーションを使い取得する

パラメータ

```
@GetMapping("/showItem/{id}")
public String show(
    @PathVariable("id") Integer id,
    Model model) {
  ...
```

この例では、("id") は省略可能
パラメータ名(青字)と引数名(緑字)
が同じ場合は記述を省略できる

URIテンプレート変数の例

- http://www.example.com/showMemberBetween/25/35
  のような複数のパラメータに対応する例

```
@GetMapping("/showMemberBetween/{from}/{to}")
public String show(
    @PathVariable("from") Integer from,
    @PathVariable("to") Integer to,
    Model model) {
   ...
```

フォワードとリダイレクト

フォワードとリダイレクト

- コントローラーメソッドの戻り値にプレフィックスを付けることで、フォワードやリダイレクトの指定ができる

フォワードの場合は、**forward:**プレフィックスを付ける

```
return "forward:/loginProcess";
```

  - フォワード先に指定したURLに対応するメソッドが実行されるが、URLは変わらない

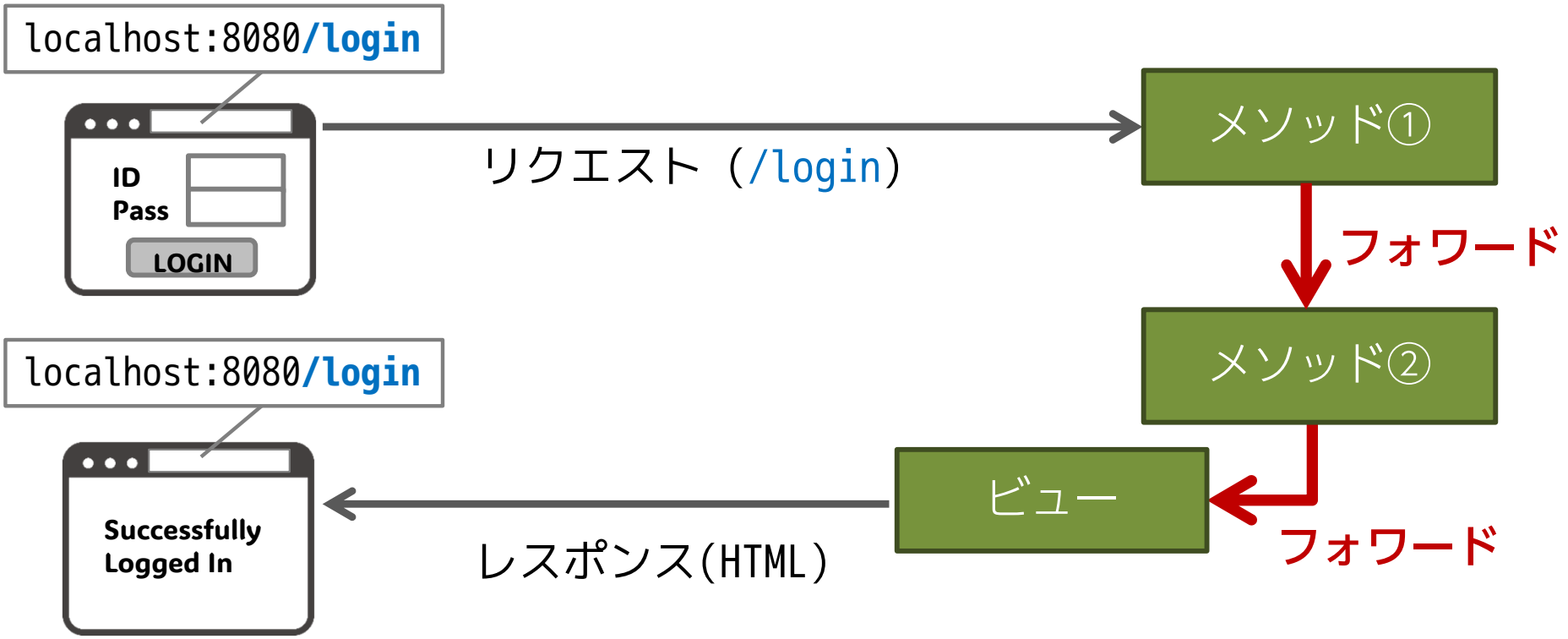
指定したURLへリダイレクトする場合は、**redirect:**プレフィックスを付ける

```
return "redirect:/login";
```

  - リダイレクトをすると、URLが変わる
    ⇒ 新たなURLに対応するメソッドが実行される
  - 外部サイトへのリダイレクトも可能

フォワード

- **フォワード**は、別のコントローラーメソッドやビューにリクエスト情報を転送する仕組み
  - ブラウザ上のURLは変わらない



フォワード先へデータを渡す

- フォワード先のコントローラーメソッドにデータを渡すには、HttpServletRequestを利用する
  - メソッドの引数として、HttpServletRequestを指定し、setModelAttribute()やgetModelAttribute()メソッドを使用する

```
@GetMapping("/showPrice")
public String show(HttpServletRequest req) {
    req.setAttribute("price", 100);
    req.setAttribute("amount", 5);
    return "forward:/calcPrice";
}

@GetMapping("/calcPrice")
public String calc(HttpServletRequest req) {
    int price = (int) req.getAttribute("price");
    int amount = (int) req.getAttribute("amount");
    req.setAttribute("total", price * amount);
    return "result";
}
```

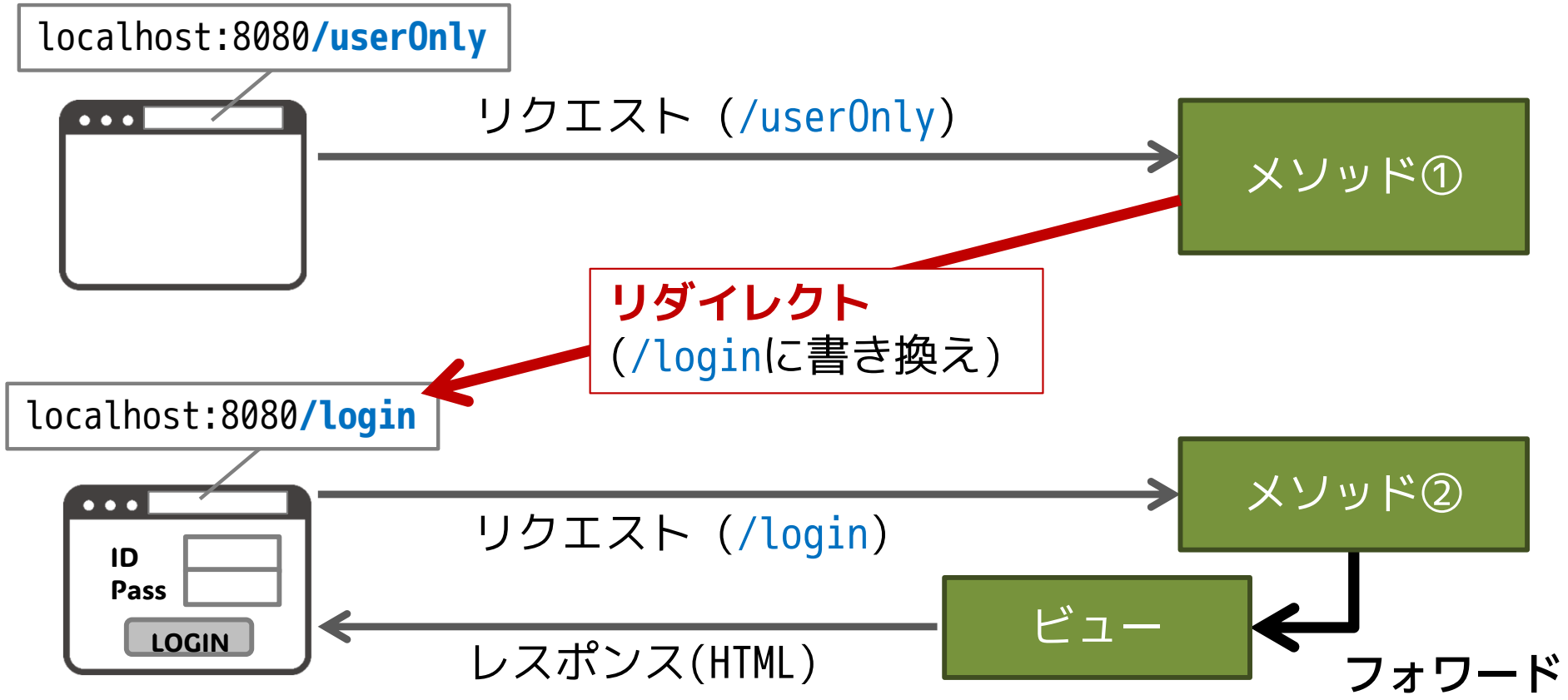
setAttribute()で保存
⇒フォワード先で利用できる

getAttribute()で取り出す
⇒戻り値はObject型なので、
必要に応じてキャストする

ビューに渡す場合は、
Model#addAttribute()でもよい

リダイレクト

- **リダイレクト**は、クライアントを別のURLに遷移させる仕組み
  - ブラウザ上のURLが遷移先ページのURLに変わる



リダイレクト先へデータを渡す

- リダイレクト後に呼び出されるコントローラーメソッドにデータを渡すには、**セッション**を利用する必要がある
  - メソッドの引数として、HttpSessionを指定する

```
@GetMapping("/showPrice")
public String show(HttpSession session) {
    session.setAttribute("price", 100);
    session.setAttribute("amount", 5);
    return "redirect:/calcPrice";
}

@GetMapping("/calcPrice")
public String calc(HttpSession session,
      Model model) {
    int price = (int) session.getAttribute("price");
    int amount = (int) session.getAttribute("amount");
    model.addAttribute("total", price * amount);
    return "result";
}
```

setAttribute()で保存
⇒リクエストをまたいで利用できる

getAttribute()で取り出す
⇒戻り値はObject型なので、
必要に応じてキャストする

セッションについては後の章で学習する

練習問題

- 練習06-4