

A faint, light blue world map is visible in the background of the slide, centered behind the text.

INTERNET ACADEMY

Institute of Web Design & Software Services

Spring Boot 4

インターネット・アカデミー

Spring Boot 4 目次

- データの検索 (SELECT)
- データの削除 (DELETE)
- データの更新 (UPDATE)

A faint, light blue world map is visible in the background of the slide, centered behind the main text.

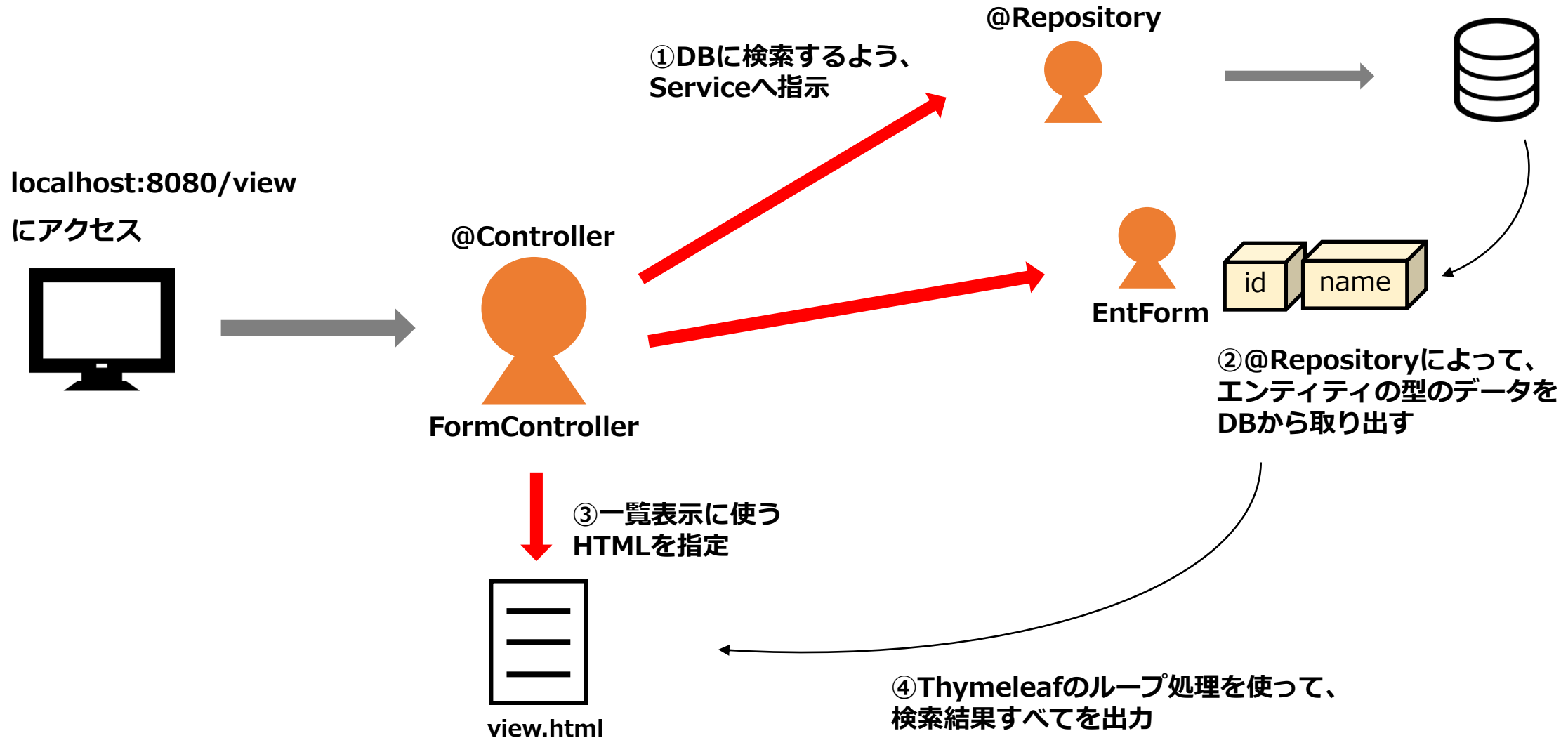
データの検索 (SELECT)

DAOパターン

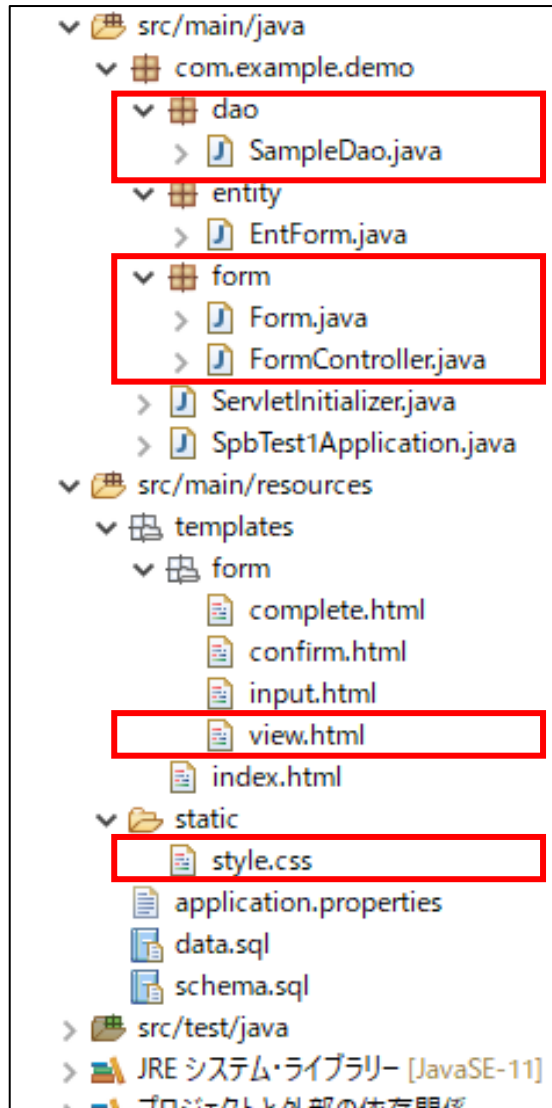
・データ検索時(SELECT)



検索時のFormControllerの動きのイメージ



追加・編集するファイルの確認



DAO

フォーム周り

HTML

CSS

データベース処理(DAO)



SampleDao.java

```
@Repository
public class SampleDaoImpl implements SampleDao{
    :
    public List<EntForm> searchDb(){
        String sql = "SELECT * FROM sample";
        //データベースから取り出したデータをresultDB1に入れる
        List<Map<String, Object>> resultDb1 = db.queryForList(sql);

        //画面に表示しやすい形のList(resultDB2)を用意
        List<EntForm> resultDb2 = new ArrayList<EntForm>();

        //1件ずつピックアップ
        for(Map<String,Object> result1:resultDb1) {
            //データ1件分を1つのまとまりとしたEntForm型の「entformdb」を生成
            EntForm entformdb = new EntForm();
            //id、nameのデータをentformdbに移す
            entformdb.setId((int)result1.get("id"));
            entformdb.setName((String)result1.get("name"));
            //移し替えたデータを持ったentformdbを、resultDB2に入れる
            resultDb2.add(entformdb);
        }
        //Controllerに渡す
        return resultDb2;
    }
}
```

データベース処理(DAO)



SampleDao.java

```
@Repository
public class SampleDaoImpl implements SampleDao{
    :
    public List<EntForm> searchDb(){
        String sql = "SELECT * FROM sample";
        ① //データベースから取り出したデータをresultDB1に入れる
        List<Map<String, Object>> resultDb1 = db.queryForList(sql);

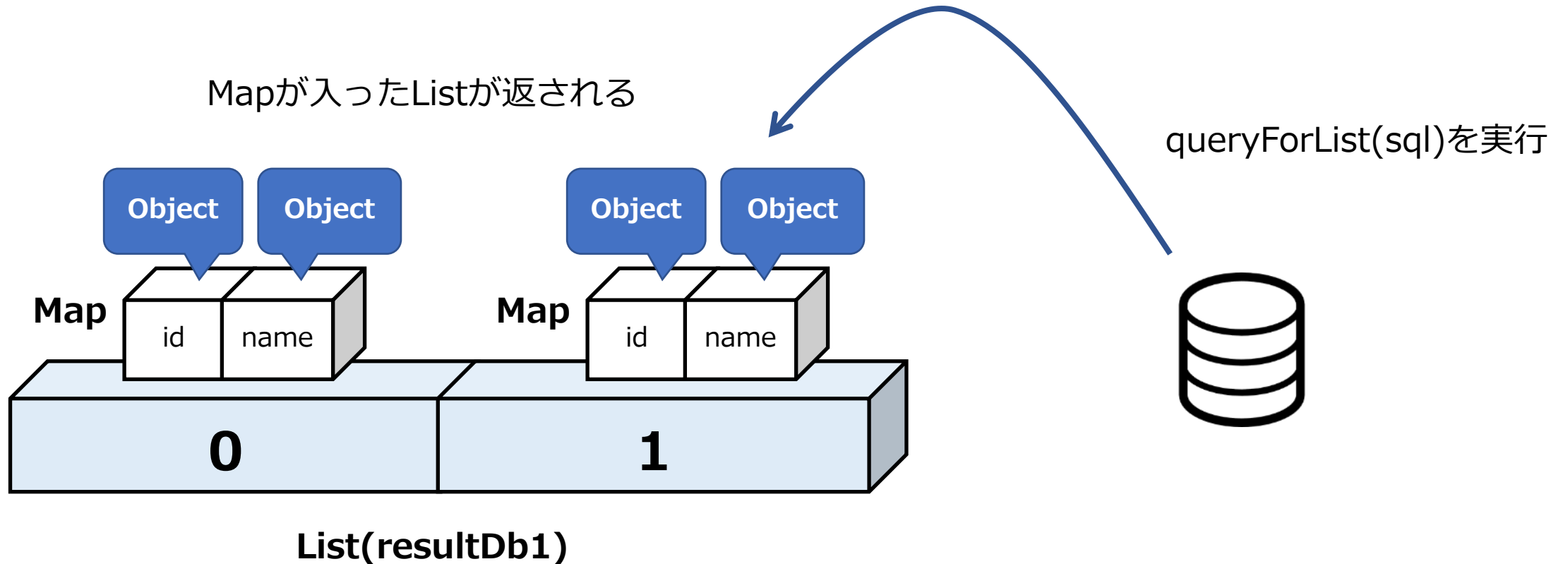
        //画面に表示しやすい形のList(resultDB2)を用意
        List<EntForm> resultDb2 = new ArrayList<EntForm>();

        //1件ずつピックアップ
        for(Map<String,Object> result1:resultDb1) {
            //データ1件分を1つのまとまりとしたEntForm型の「entformdb」を生成
            EntForm entformdb = new EntForm();
            //id、nameのデータをentformdbに移す
            entformdb.setId((int)result1.get("id"));
            entformdb.setName((String)result1.get("name"));
            //移し替えたデータを持ったentformdbを、resultDB2に入れる
            resultDb2.add(entformdb);
        }
        //Controllerに渡す
        return resultDb2;
    }
}
```

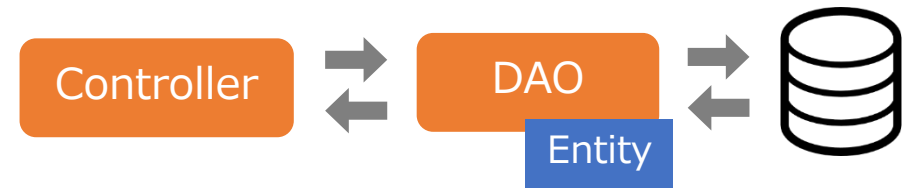

データベース処理(DAO)

① の解説

```
List<Map<String,Object>> resultDb1 = db.queryForList(sql);
```



データベース処理(DAO)

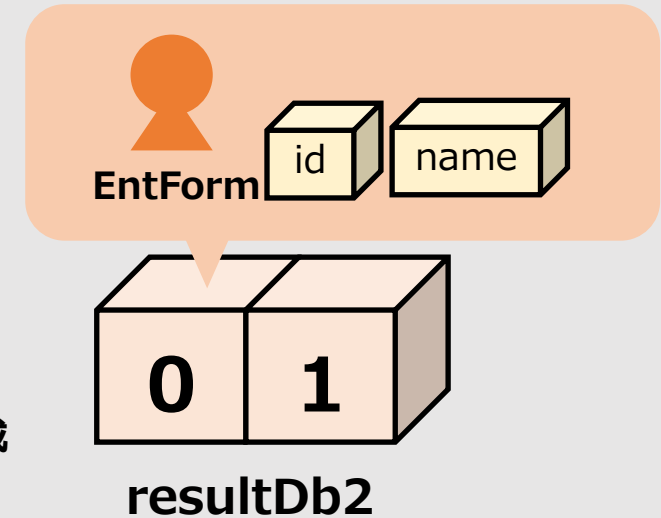


SampleDao.java

```
@Repository
public class SampleDaoImpl implements SampleDao{
    :
    public List<EntForm> searchDb(){
        String sql = "SELECT * FROM sample";
        //データベースから取り出したデータをresultDB1に入れる
        List<Map<String, Object>> resultDb1 = db.queryForList(sql);

        ② //画面に表示しやすい形のList(resultDB2)を用意
        List<EntForm> resultDb2 = new ArrayList<EntForm>();

        //1件ずつピックアップ
        for(Map<String,Object> result1:resultDb1) {
            //データ1件分を1つのまとまりとしたEntForm型の「entformdb」を生成
            EntForm entformdb = new EntForm();
            //id、nameのデータをentformdbに移す
            entformdb.setId((int)result1.get("id"));
            entformdb.setName((String)result1.get("name"));
            //移し替えたデータを持ったentformdbを、resultDB2に入れる
            resultDb2.add(entformdb);
        }
        //Controllerに渡す
        return resultDb2;
    }
}
```



データベース処理(DAO)



SampleDao.java

```
@Repository
public class SampleDaoImpl implements SampleDao{
    :
    public List<EntForm> searchDb(){
        String sql = "SELECT * FROM sample";
        //データベースから取り出したデータをresultDB1に入れる
        List<Map<String, Object>> resultDb1 = db.queryForList(sql);

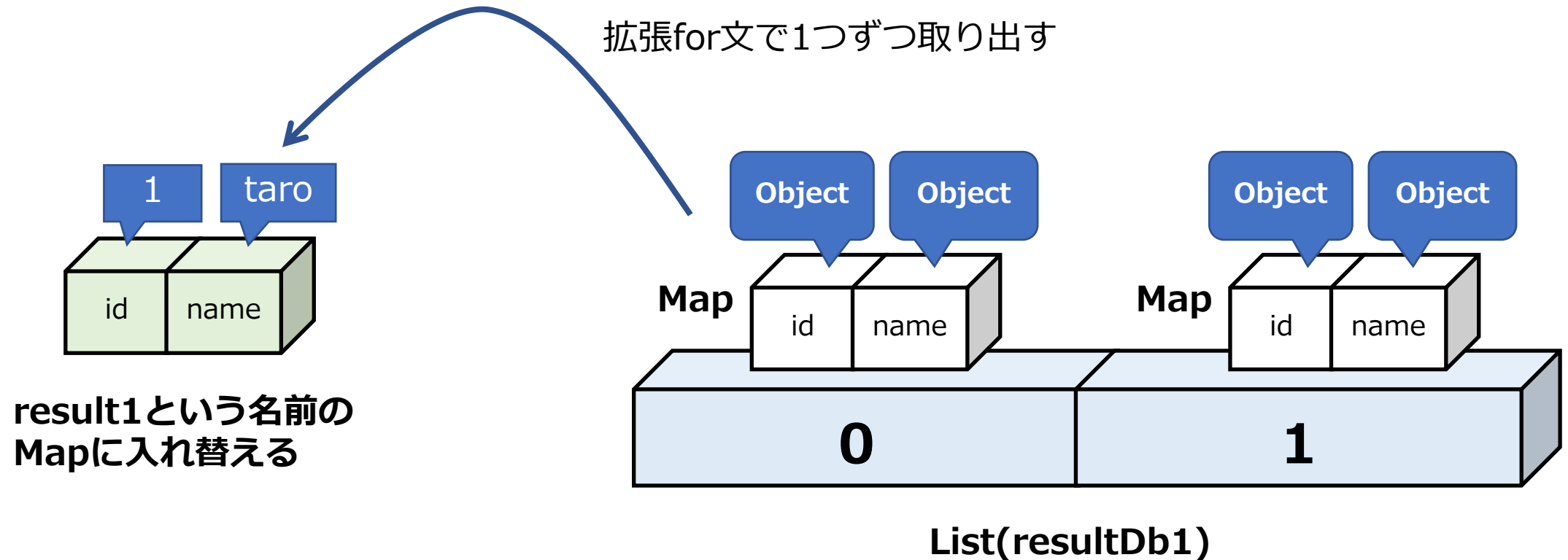
        //画面に表示しやすい形のList(resultDB2)を用意
        List<EntForm> resultDb2 = new ArrayList<EntForm>();

        ③ //1件ずつピックアップ
        for(Map<String,Object> result1:resultDb1) {
            //データ1件分を1つのまとまりとしたEntForm型の「entformdb」を生成
            EntForm entformdb = new EntForm();
            //id、nameのデータをentformdbに移す
            entformdb.setId((int)result1.get("id"));
            entformdb.setName((String)result1.get("name"));
            //移し替えたデータを持ったentformdbを、resultDB2に入れる
            resultDb2.add(entformdb);
        }
        //Controllerに渡す
        return resultDb2;
    }
}
```

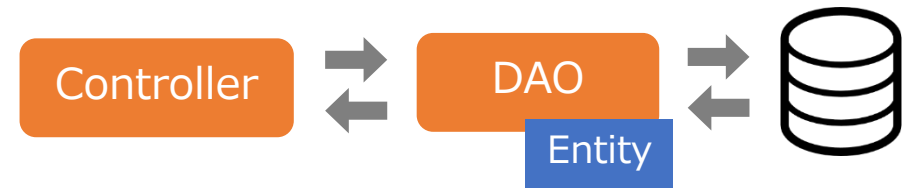
データベース処理(DAO)

③ の解説

```
for(Map<String, Object> result1 : resultDb1) { . . . }
```



データベース処理(DAO)



SampleDao.java

```
@Repository
public class SampleDaoImpl implements SampleDao{
    :
    public List<EntForm> searchDb(){
        String sql = "SELECT * FROM sample";
        //データベースから取り出したデータをresultDB1に入れる
        List<Map<String, Object>> resultDb1 = db.queryForList(sql);

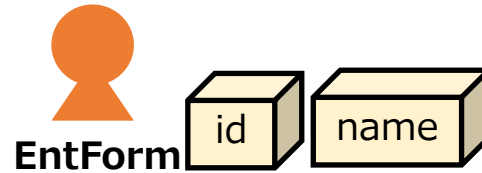
        //画面に表示しやすい形のList(resultDB2)を用意
        List<EntForm> resultDb2 = new ArrayList<EntForm>();

        //1件ずつピックアップ
        for(Map<String,Object> result1:resultDb1) {
            //データ1件分を1つのまとまりとしたEntForm型の「entformdb」を生成
            EntForm entformdb = new EntForm();
            //id、nameのデータをentformdbに移す
            entformdb.setId((int)result1.get("id"));
            entformdb.setName((String)result1.get("name"));
            //移し替えたデータを持ったentformdbを、resultDB2に入れる
            resultDb2.add(entformdb);
        }
        //Controllerに渡す
        return resultDb2;
    }
}
```

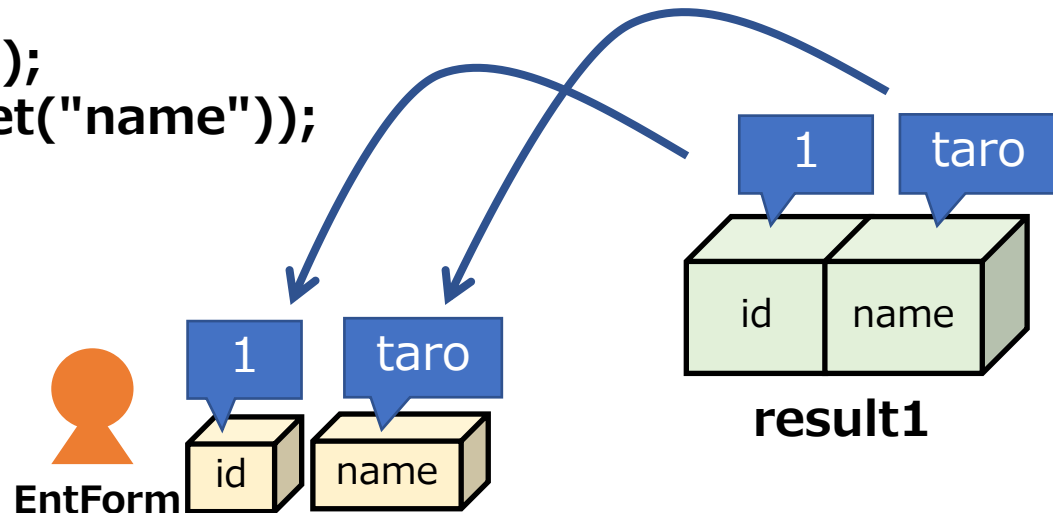
データベース処理(DAO)

④ の解説

//データ1件分を1つのまとまりとしたEntForm型の「entformdb」を生成
`EntForm entformdb = new EntForm();`

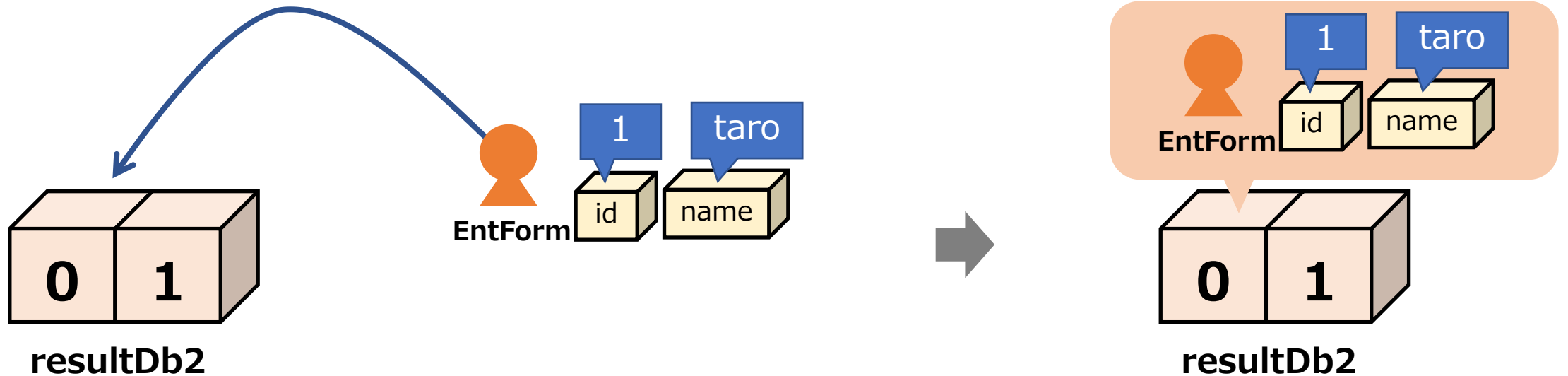


//id、nameのデータをentformdbに移す
`entformdb.setId((int)result1.get("id"));`
`entformdb.setName((String)result1.get("name"));`



データベース処理(DAO)

```
//移し替えたデータを持ったentformdbを、resultDB2に入れる  
resultDb2.add(entformdb);
```



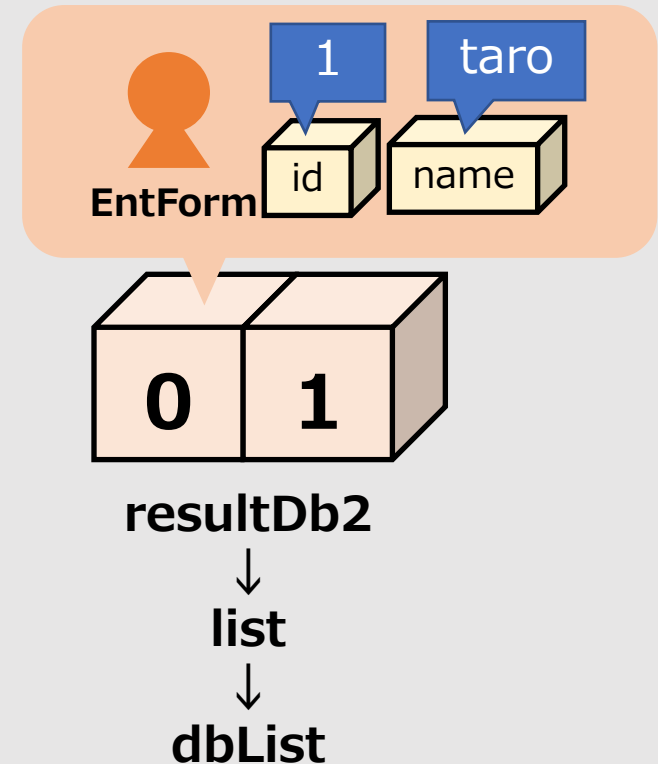
データベースから取り出した1行分のデータが、
「エンティティの状態になって、ArrayListに格納されている」
ところがポイント！

データベース処理(Controller)

FormController.javaに追記

```
@Controller
public class FormController {
    :
    :
    //全件検索(SELECT)
    @RequestMapping("/view")
    public String view(Model model) {
        List<EntForm> list = sampledao.searchDb();
        model.addAttribute("dbList",list);

        model.addAttribute("title","一覧ページ");
        return "form/view";
    }
}
```

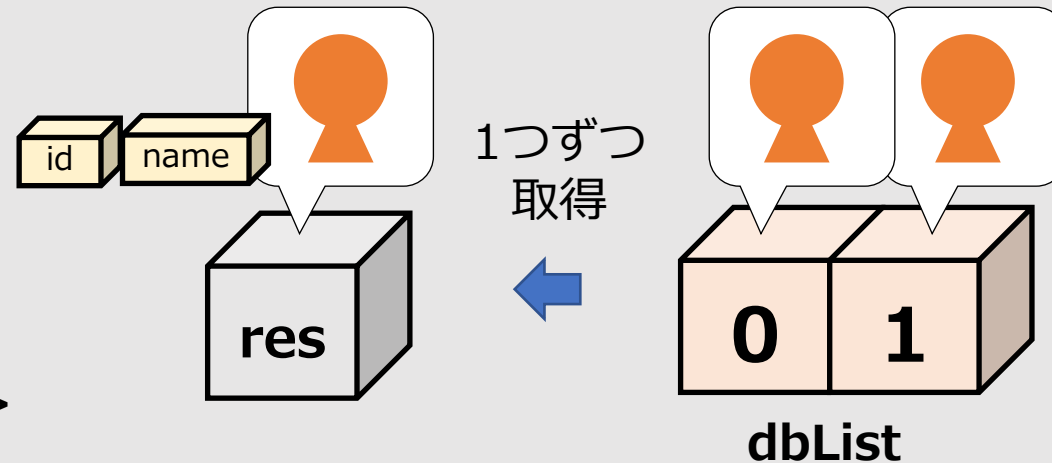


※名前は変わるが内容は同じ

Viewの作成 (view.html)

view.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Hello</title>
<meta charset="utf-8" />
<link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<h1 th:text="${title}"></h1>
<table>
<tr>
<th>id</th>
<th>名前</th>
</tr>
<tr th:each="res : ${dbList}">
<td th:text="${res.id}">1</td>
<td th:text="${res.name}">名前</td>
</tr>
</table>
<p><a href="/form" th:href="@{/form}">フォーム入力画面へ</a></p>
</body>
</html>
```



Viewの作成 (style.css)

style.css

```
@charset "UTF-8";

table {
  border-collapse: collapse;
}

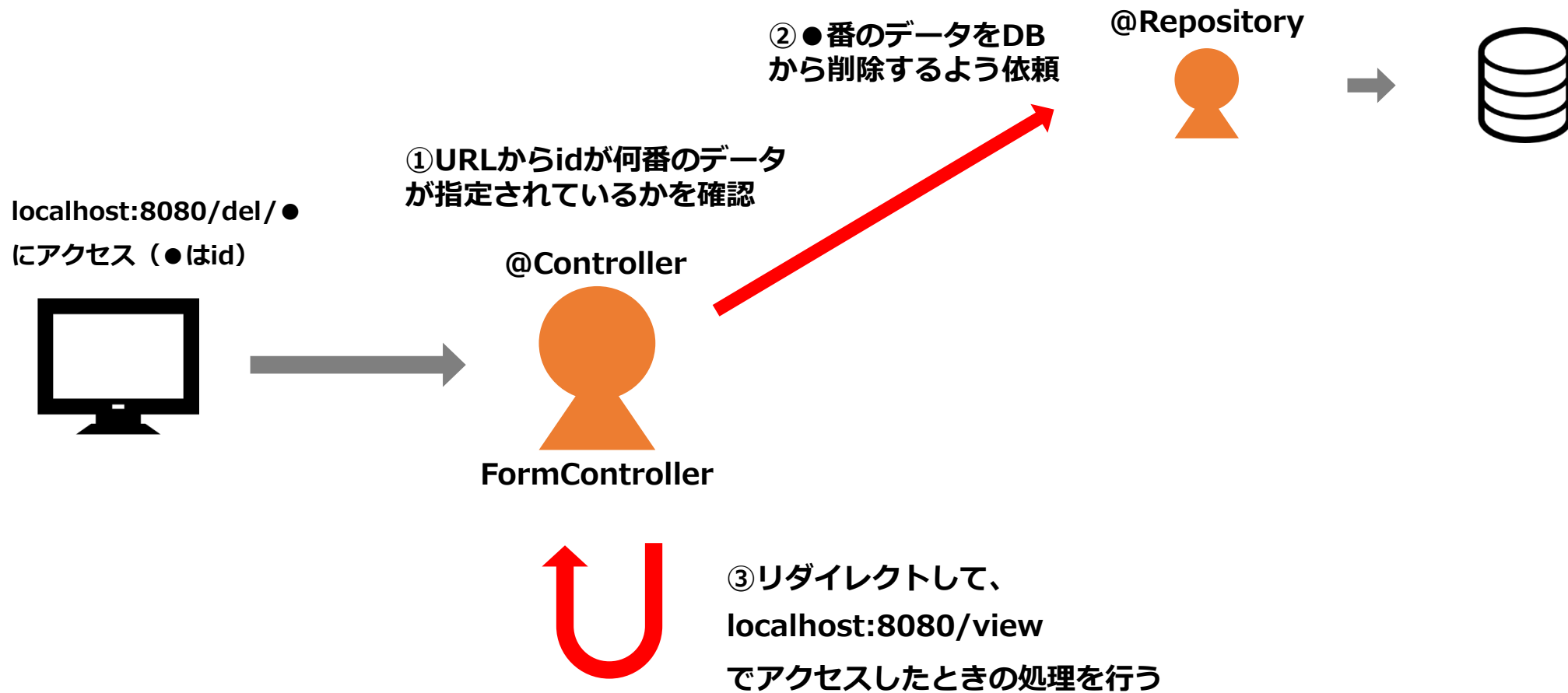
table th {
  background: #379ce8;
  border: solid 1px #ccc;
  color: #fff;
  padding: 10px;
}

table td {
  border: solid 1px #ccc;
  padding: 10px;
}
```

A faint, light blue world map is visible in the background of the slide, centered behind the main text.

データの削除 (DELETE)

FormControllerの動きのイメージ



データベースの削除(View)

view.htmlに追加

```
<table>
  <tr>
    <th>id</th>
    <th>名前</th>
    <th> </th>
  </tr>
  <tr th:each="res : ${dbList}">
    <td th:text="${res.id}">1</td>
    <td th:text="${res.name}">名前</td>
    <td>
      <form th:action="@{/del/{id}(id=${res.id})}">
        <input type="submit" value="削除" />
      </form>
    </td>
  </tr>
</table>
```


idが1のとき「localhost:8080/del/1」になる

※@{/del/\${res.id}}は文法エラー

データベースの削除(Controller)

FormController.javaに追加

```
@Controller
public class FormController {
    :
    :
    //削除(DELETE)
    @RequestMapping("/del/{id}")
    public String destory(@PathVariable Long id) {
        sampledao.deleteDb(id);
        return "redirect:/view";
    }
}
```

 @PathVariableでURLの数字を、変数に受け取っている

データベースの削除(DAO)

SampleDao.javaに追加

```
:
@Repository
public class SampleDao{
    :
    :
    //削除(DELETE)
    public void deleteDb(Long id) {
        //コンソールに表示
        System.out.println("削除しました");
        //DBからデータを削除
        db.update("delete from sample where id=?", id);
    }
}
```

コンソール出力 (確認のため)



データの更新 (UPDATE)

編集の遷移イメージ

一覧画面

id	name	-	-
1	Taro	編集	削除
2	Jiro	編集	削除

localhost:8080/view

ボタンを押して
localhost:8080/edit/1
にアクセス



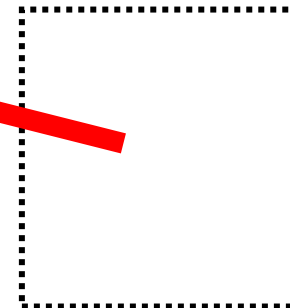
編集画面

id 1

name

localhost:8080/edit/1

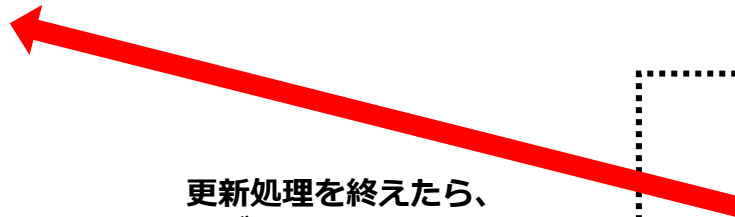
ボタンを押して
localhost:8080/edit/1/exe
にアクセス



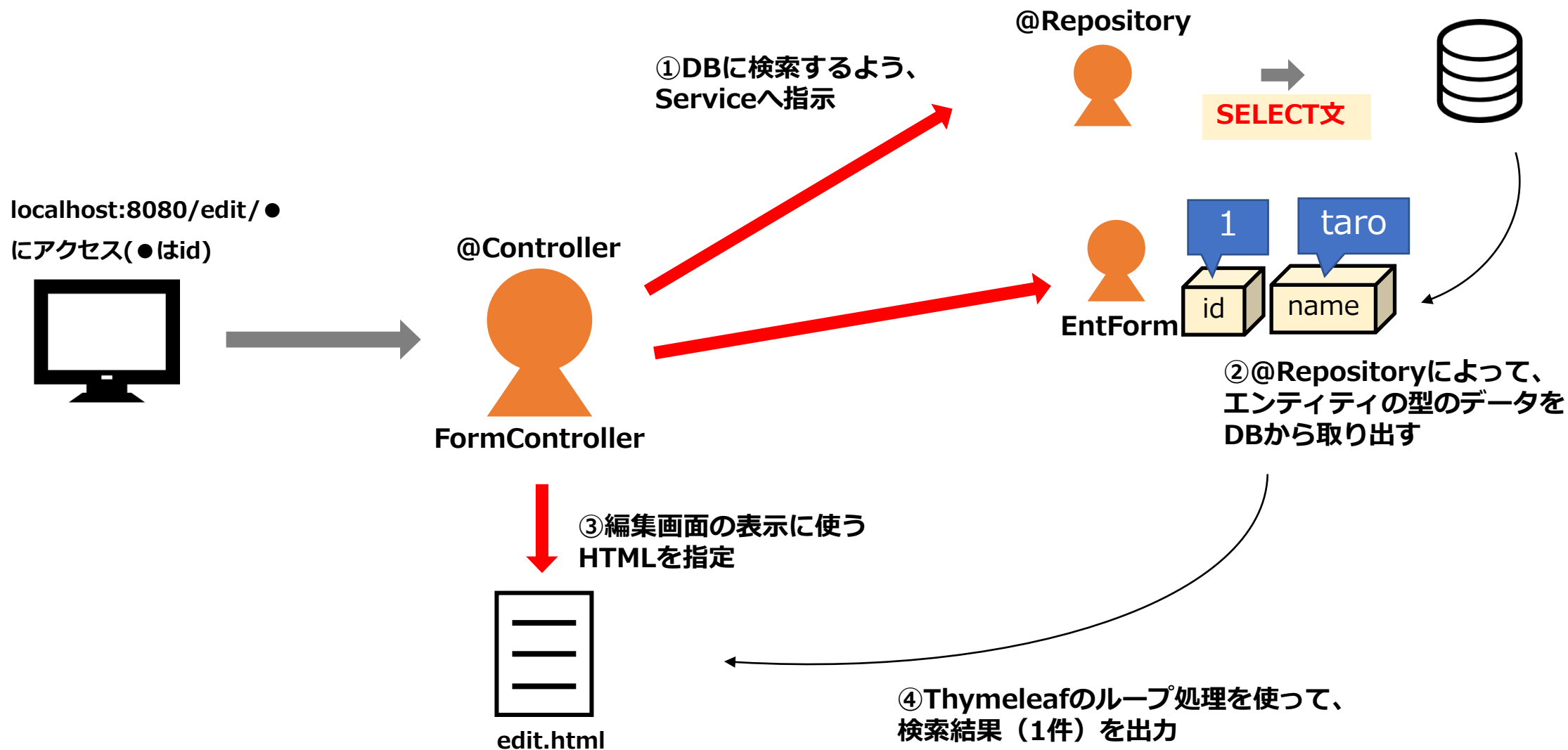
localhost:8080/edit/1/exe

※画面は出ない

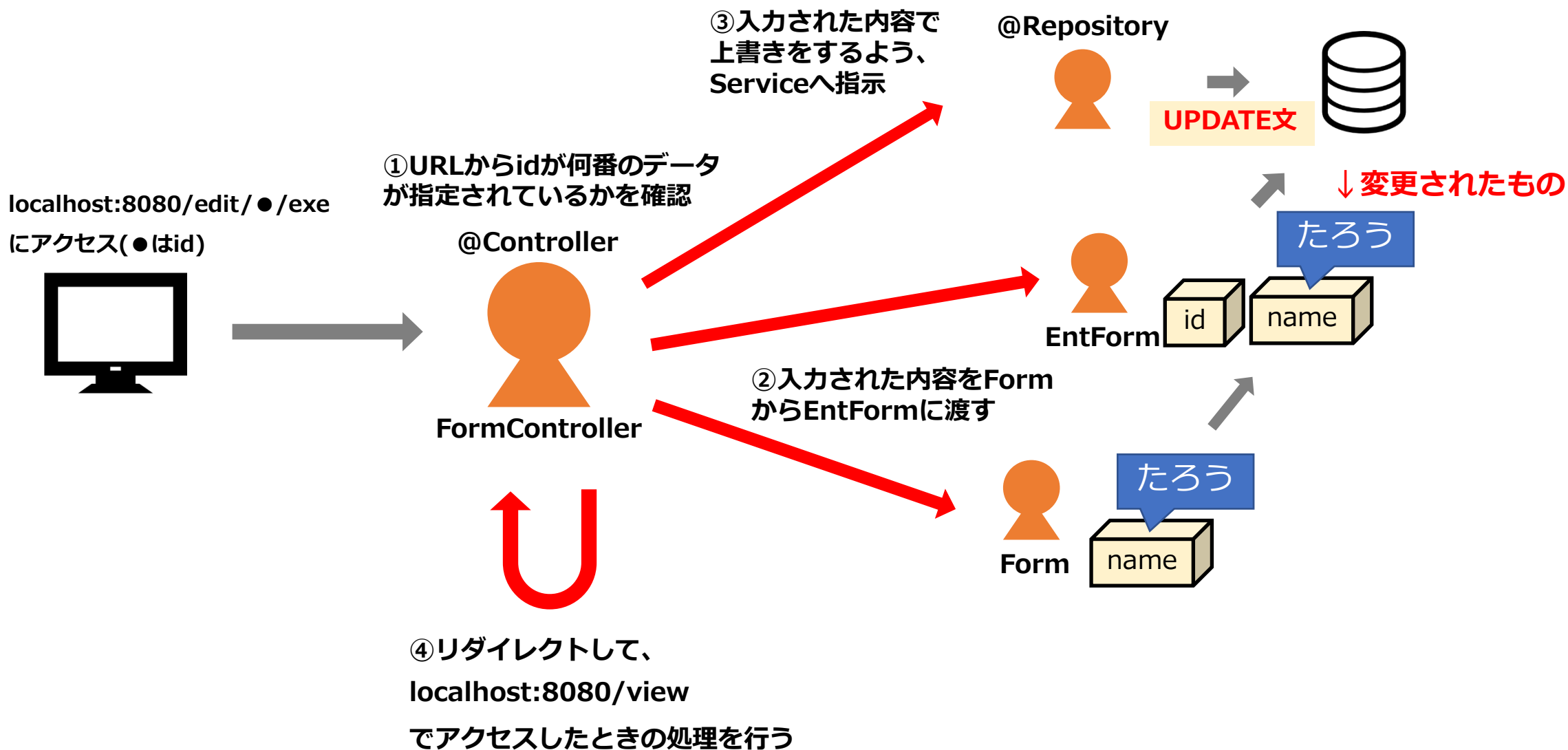
更新処理を終えたら、
リダイレクトする



FormControllerの動きのイメージ



FormControllerの動きのイメージ



演習

以下の名前でそれぞれ指定して、更新処理を行う機能を実装してみましょう

処理内容	URL	Controller	DAO (Repository)	SQL文	戻り値
編集データの 表示	/edit/●	editView()	selectOne()	SELECT	エンティティ (EntForm)
データの更新	/edit/●/exe	editExe()	updateDb()	UPDATE	なし

！ポイント

これまでのSQL操作と近いため、ほぼコピー＆ペーストで作れます

データの更新の解答

- **STEP 1**

編集画面を表示する

- **STEP 2**

更新処理を実行する

データベースの更新(1 : 更新画面を表示)

view.htmlに追加

```
<table>
  <tr>
    <th>id</th>
    <th>名前</th>
    <th></th>
    <th></th>
  </tr>
  <tr th:each="res:${dbList}">
    <td th:text="${res.id}">ID</td>
    <td th:text="${res.name}">名前</td>
    <td>
      <form th:action="@{/edit/{id}(id=${res.id})}">
        <input type="submit" value="編集">
      </form>
    </td>
    <td>
      <form th:action="@{/del/{id}(id=${res.id})}">
        <input type="submit" value="削除">
      </form>
    </td>
  </tr>
</table>
```

データベースの更新(1 : 更新画面を表示)

edit.htmlを追加

```
:
<body>
  <h1 th:text="${title}"></h1>

  <div th:object="${form}">
    <form action="#" method="get" th:action="@{/edit/___*{id}___/exe}">
      <p>
        番号:<span th:text="*{id}"></span>
        <input type="hidden" name="id" th:value="*{id}">
      </p>
      <p>
        名前: <input type="text" name="name" th:value="*{name}">
      </p>
      <input type="submit" value="更新">
    </form>
  </div>
</body>
:
```

データベースの更新(1 : 更新画面を表示)

FormController.javaに追加

```
@Controller
public class FormController {
    :
    :
    //更新画面の表示(SELECT)
    @RequestMapping("/edit/{id}")
    public String editView(@PathVariable Long id, Model model) {

        //DBからデータを1件取ってくる(リストの形)
        List<EntForm> list = sampledao.selectOne(id);

        //リストから、オブジェクトだけをピックアップ
        EntForm entformdb = list.get(0);

        //スタンバイしているViewに向かって、データを投げる
        model.addAttribute("form", entformdb);
        model.addAttribute("title", "編集ページ");
        return "form/edit";
    }
}
```


データベースの更新(1 : 更新画面を表示)

SampleDao.javaに追加

```
//更新画面の表示(SELECT)
public List<EntForm> selectOne(Long id) {
    //コンソールに表示
    System.out.println("編集画面を出します");

    //データベースから目的の1件を取り出して、そのままresultDB1に入れる
    List<Map<String, Object>> resultDb1 = db.queryForList("SELECT * FROM sample where id=?", id);

    //画面に表示しやすい形のList(resultDB2)を用意
    List<EntForm> resultDb2=new ArrayList<EntForm>();

    //1件ずつピックアップ
    for(Map<String,Object> result1:resultDb1) {
        //データ1件分を1つのまとまりとするので、EntForm型の「entformdb」を生成
        EntForm entformdb = new EntForm();
        //id、nameのデータをentformdbに移す
        entformdb.setId((int)result1.get("id"));
        entformdb.setName((String)result1.get("name"));
        //移し替えたデータを持ったentformdbを、resultDB2に入れる
        resultDb2.add(entformdb);
    }
    //Controllerに渡す
    return resultDb2;
}
```

データベースの更新(2 : 更新処理の実行)

edit.htmlの確認 ※確認のみで変更はありません

```
:
<body>
  <h1 th:text="${title}"></h1>

  <div th:object="${form}">
    <form action="#" method="get" th:action="@{/edit/___*{id}___/exe}">
      <p>
        番号:<span th:text="*{id}"></span>
        <input type="hidden" name="id" th:value="*{id}">
      </p>
      <p>
        名前: <input type="text" name="name" th:value="*{name}">
      </p>
      <input type="submit" value="更新">
    </form>
  </div>
</body>
:
```

データベースの更新(1 : 更新画面を表示)

FormController.javaに追加

```
@Controller
public class FormController {
    :
    //更新処理(UPDATE)
    @RequestMapping("/edit/{id}/exe")
    public String editExe(@PathVariable Long id, Model model, Form form) {

        //フォームの値をエンティティに入れ直し
        EntForm entform = new EntForm();
        entform.setName(form.getName());
        //更新の実行
        sampledao.updateDb(id,entform);

        //一覧画面へリダイレクト
        return "redirect:/view";
    }
}
```

データベースの更新(1 : 更新画面を表示)

SampleDao.javaに追加

@Repository

```
public class SampleDao{
```

```
    :
```

```
    :
```

```
    //更新の実行(UPDATE)
```

```
    public void updateDb(Long id, EntForm entform) {
```

```
        //コンソールに表示
```

```
        System.out.println("編集の実行");
```

```
        //UPDATEを実行
```

```
        db.update("UPDATE sample SET name = ? WHERE id = ?",entform.getName(), id);
```

```
    }}
```