# Controllerの実装

本ページの章立ては記述速度重視で適当になっています。（今後、大きく変更する可能性があります。）

## リダイレクト先URLの指定方法

### 同じWebアプリ内で遷移する場合

```
@PostMapping("hello")
public String hello(RedirectAttributes redirectAttrs) {
    String id = "aaaa";
        redirectAttrs.addAttr("id", id);
    return "redirect:/sample/hello"; // <1>
}
```

- <1> redirect:プレフィックスを付ける。
  - リダイレクト先のURLは/始まりでコンテキストルートからのパスを記述すること。/始まりではないパスにはコンテキストパスが補完されない。

### 外部URLへ遷移する場合

```
@PostMapping("hello")
public String hello(RedirectAttributes redirectAttrs) {
    String q = "aaaa";
        redirectAttrs.addAttr("q", q);
    return "redirect:https://www.google.co.jp/search"; // <1>
}
```

- <1> redirect:プレフィックスを付け、URLは/始まりではなく、絶対パスで記述する。

上記のコード例ではhttps://www.google.co.jp/search?q=aaaaにリダイレクトされることになる。

## リダイレクト先にデータを渡す方法

次の3つのパターンについて説明する。

- リクエストパラメータとして渡す
- リダイレクト先URLのパスパラメータとして渡す
- リダイレクト先に構造データ(Formクラス)を渡す

### リクエストパラメータとして渡す

```
@PostMapping("hello")
public String hello(RedirectAttributes redirectAttrs) // <1>
    String id = "aaaa";
        redirectAttrs.addAttr("id", id)// <2>
    return "redirect:/sample/hello"; // <3>
}
```

- <1> RedirectAttributes を引数に指定する
- <2> RedirectAttributesのaddAttributeメソッドでリクエストパラメータを指定する
  - RedirectAttributesで追加したリクエストパラメータは適切にURLエンコードされる
- <3> "redirect:/sample/hello?id=" + id というように動的なパラメータを使ってクエリ文字列を構築するのは禁止
  - URLエンコードなど考慮しなくてはならないため

上記のコード例では/sample/hello?id=aaaaにリダイレクトされることになる。

### リダイレクト先URLのパスパラメータとして渡す

```
@PostMapping("hello")
public String hello(RedirectAttributes redirectAttrs)// <1>
    String id "aaaa";
        redirectAttrs.addAttr"id", id)// <2>
    return "redirect:/sample/hello/{id}"; // <3>
}
```

- <1> RedirectAttributes を引数に指定する
- <2> RedirectAttributesのaddAttributeメソッドでパスパラメータを指定する
  - RedirectAttributesで追加したパスパラメータは適切にURLエンコードされる
- <3> URL文字列の中にプレースホルダ{name}を埋めると、RedirectAttributesで追加したパラメータを適切に埋められる

上記のコード例では/sample/hello/aaaaにリダイレクトされることになる。

### リダイレクト先に構造データ(Formクラス)を渡す

```
@PostMapping("hello")
public String hello(RedirectAttributes redirectAttrs)// <1>
        HelloForm helloFnew HelloForn"Bean Hello Wor");!
        redirectAttrs.addFlashAttribute(helloF(// <2>
    return "redirect:/sample/hello";
}

@GetMapping("hello")
public String helloComplete(HelloForm helloForm)// <3>
    return "sample/complete";
}
```

- <1> RedirectAttributes を引数に指定する
- <2> RedirectAttributesのaddFlashAttributeメソッドでリダイレクト先に渡すFormオブジェクトを指定する
- <3> リダイレクト先のControllerでは引数に同じFormクラスを指定して、データを取得する

上記のコード例では/sample/helloにリダイレクトされることになる。

RedirectAttributesのaddFlashAttributeメソッドで指定されたデータは内部的にHTTPセッションに追加され、
リダイレクト先のControllerメソッドが呼び出される時点で、HTTPセッションから削除されます。
つまり、HTTPセッションでデータを保持するため、渡すデータはjava.io.Serializableを実装していること。

参考

- 3.4.1.4.6. リダイレクト先にデータを渡す — 3.4. アプリケーション層の実装 — TERASOLUNA Server Framework for Java (5.x) Development Guideline 5.3.0.RELEASE documentation

## ControllerHelperクラスの実装方法

Controllerをサポートするクラスはサポート対象のControllerと同一パッケージに「{サポートするControllerクラス名}Helper」というクラス名で作成する。
複数のControllerをサポートする場合は、なるべくサポート対象Controllerに近いパッケージに「{任意の名前}ControllerHelper」というクラス名で作成する。(安易にcommonパッケージに配置しないこと)

```
@Component // <1>
public class FooControllerHelper {

    @Autowired
    private FooSessionDto// <2>

    public FooDto procF(String bar) {
        // ...
        }
```

}

- <1> @Componentを付与し、コンポーネントスキャン対象にする
- <2> Spring管理対象なのでAutowired可能

ControllerHelperはControllerでServiceなどと同様にAutowiredでインジェクションして使用する。