6-3.Springboot基礎編

SpringBootの仕組み

前ページの解説をします。
まず、HelloSpringBootWebControllerクラスのindexメソッドを見てみましょう！

```
    @RequestMapping(value="/", method=RequestMethod.GET)
    public ModelAndView index(ModelAndView mv) {
```

リクエストマッピング

index メソッドに、 `@RequestMapping` アノテーションが設定されています。これは **リクエストマッピング** と呼ばれる機能のアノテーションです。
リクエストマッピングとは、「このアドレスにアクセスすると呼び出されるメソッドを実行する」という、アクセスするアドレスと実行する処理（indexメソッド）の結びつきを設定します。

ここでは、`@RequestMapping`アノテーション の引数に、`value="/"` と `method=RequestMethod.GET` を指定しています。
value 属性は `http://oo/` にアクセスした時に、このメソッドが実行されるようになります。
method 属性は、GETメソッドでのアクセス時、POSTメソッドでのアクセス時なのかを指定することができます。

以上のことから、URLにアクセスしたいときに処理したいメソッドを以下のように指定してもメソッドが実行されないことを覚えておきましょう。

```
@RequestMapping(value="/access", method=RequestMethod.POST) を指定した場合


// GET送信でのアクセスなので実行されない
http://localhost:8080/access

// value属性と合致しないため実行されない
http://localhost:8080/abc
```

ModelAndView について

ModelAndView とはレスポンスとして返せるクラスです。その名の通り「モデル」と「ビュー」は、MVCの「 **Model** 」と「 **View** 」のことで、データを管理するモデルと、画面表示に関するビューをまとめて扱います。
このクラスを使用することで表示させたい **View** を指定することができます。
それが`setViewName`メソッドです。

`setViewName` メソッドの引数に表示させたいビューの名前を設定します。これにより **templates** フォルダから、引数に指定した名前のテンプレートをロードするようになります。
ここでは `"index"` としているので **templates** フォルダ内から **index.html** という名前のファイルをテンプレートとして読み込むようになります。
（拡張子は指定する必要はありませんん。"index" で自動的にindex.htmlが読み込まれます）。

最後に return文 で **ModelAndView** を返してあげます。
ModelAndView を返すことによって、 **View** に渡したい情報を一緒に返すことができます。

ただ、 return文 で返すオブジェクトは ModelAndView だけではなく、文字列の場合もあります。
下記ソースコードは index メソッドと同じ働きをします。

```
    @RequestMapping(value="/", method=RequestMethod.GET)
    public String sample() {      // 文字列を返すため 戻り値の型は String型にすること
        return "index";
    }
```

ModelAndview を返さず、文字列を返した場合は、その文字列と同じ名前のテンプレートをロードするようになります。
ModelAndView は ロードするテンプレートを指定する `setViewName` メソッドだけでなく コントローラークラスからViewに値を渡すことができたりするため、
今回の indexメソッドのように、「 `http://localhost:8080/` にアクセスしたときに、 index.html をロードする」といった、単純な処理をしたい場合は文字列を返した方が簡単に記述することが出来ます。

以上のように コントローラークラス の基本的な流れは

?

1. @RequestMapping で、リクエストマッピングや、GET送信,POST送信 を指定

2. ModelAndView で、コントローラークラスからView に渡したい情報を指定
3. retuen文 で、View の遷移先を決定

となります。この基本さえ覚えておけば大丈夫です！

@RequestParam について

次にsendメソッドを見てみましょう！

```
    @RequestMapping(value="/result", method=RequestMethod.POST)
    public ModelAndView send(@RequestParam("inputvalue")String inputvalue, ModelAndView mv) {
        mv.setViewName("result");
        mv.addObject("message", inputvalue);
        return mv;
    }
```

sendメソッドの引数に **@RequestParam** アノテーションが設定されています。
このアノテーションは、リクエストに渡されたパラメータの値を示します。
`@RequestParam("inputvalue")` と指定しているので フォームから送られた値がこの引数に設定されるようになります。
今回の例であれば、 **index.html** に `<input type="text" name="inputvalue"/>` の入力フィールドに書かれた値が、`String inputvalue` に渡されるようになります。
フォームから送信された値は、このように「@RequestParamアノテーションを付けた引数」に自動的に渡されます。後は、この値を取り出して処理すればよく、特殊なオブジェクトからパラメータの値を取り出すことなく簡単に値を取り出すことができます。

addObjectメソッド について

●コントローラー側の処理
「addObject」は、 **ModelAndView** に値を保管するものです。
第1引数に名前を指定し、第2引数に保管するオブジェクトを指定します。ここでは、"message" という名前で `inputvalue` (ビューで入力された値)が保管されたことがわかります。
`setViewName` メソッドで result を指定しているので、 **result.html** に、message という名前で `inputvalue` が渡されます。

●値の出力
addObject メソッドによって **ModelAndView** に保管された値は、式言語を使って簡単に出力することができます。
result.html の以下の部分です。

result.html

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

このように、 **addObject** メソッドにより渡されたオブジェクトを出力するには `th:text="${オブジェクト名}"` と記述します。

まとめ

Springbootでのシステムの実装について簡単に説明しました。
おそらくカリキュラム上級で勉強した、 **サーブレット** を使用するよりも簡単に実装することができると感じたのではないでしょうか。
しかし、フレームワークは簡単に記述することが出来る反面、デメリットもあります。
このデメリットをしっかり把握し、学習することが近道かもしれません。

また、Springboot はアノテーションを用いて記述するフレームワークです。
覚える必要があるアノテーションや、クラスがたくさんありますが、Javaの開発現場では多く用いられているフレームワークになるので、覚えていきましょう！！

最終更新日時: 2022年 09月 10日(土曜日) 09:26