PARAMS

#01. 프로젝트 구성

1. 패키지 정보

`kr.hossam.params`

항목	설정 내용
GroupId	`kr.hossam`
ArtifactId	`params`

2. 의존성 설정

프로젝트 생성 과정에서 `dependencies` 항목에 대해 아래의 항목을 선택

항목 이름	구분
Spring Boot DevTools	기존 사용
Spring Boot Acturator Ops	기존 사용
Spring Web	기존 사용
Thymeleaf Template Engines	기존 사용
Lombok	기존 사용

3. 추가 설정

1) logback 설정파일 추가

`/src/main/resources/logback-spring.xml` 파일을 추가

(이전 수업에서 생성한 파일을 재사용)

2) UserAgent 라이브러리 설정

`build.gradle`에 의존성 설정 추가

```
implementation 'com.github.ua-parser:uap-java:1.6.1'
```

3) 기본 패키지 추가

패키지	기능
`kr.hossam.params.controllers`	컨트롤러 클래스

패키지	기능
kr.hossam.params.helpers	헬퍼 클래스
kr.hossam.params.interceptors	인터셉터 클래스

> helper 패키지 에는 UtilHelper를 포함합니다.

3. 파일 구성

```
└main
    ├java
    │  └kr
    │      └hossam
    │          └params
    │              │  MyWebConfig.java          (이전에 작성한 파일 재사용)
    │              │  ParamsApplication.java    (기본파일, 수정 없음)
    │              │
    │              ├controllers
    │              │      MyController.java     (신규, step1)
    │              │      GetController.java    (신규, step2)
    │              │      PostController.java   (신규, step3)
    │              │      PathController.java   (신규, step4)
    │              │      SurveyController.java (신규, step5)
    │              │
    │              ├helpers
    │              │      UtilHelper.java               (이전에 작성한 파일 재사
용)
    │              │
    │              └interceptors
    │                     MyInterceptor.java           (이전에 작성한 파일 재사
용)
    │
    └resources
        │  application.properties       (기본파일, 수정없음)
        │  logback-spring.xml           (이전에 작성한 파일 재사용)
        │
        ├static
        │  │  favicon.ico               (이전에 작성한 파일 재사용)
        │  │  robots.txt                (이전에 작성한 파일 재사용)
        │  │
        │  └assets
        │      └css
        │              header.css       (신규)
        │
        └templates
            │  index.html               (신규, step1)
            │
            ├fragments
            │      header.html          (신규, step1)
            │
            ├get
```

```
│      home.html            (신규, step2)
│      result.html          (신규, step2)
│
├─post
│      home.html            (신규, step3)
│      result.html          (신규, step3)
│
├─path
│      home.html            (신규, step4)
│      result.html          (신규, step4)
│
└─survey
        index.html          (신규, step5)
        step1.html          (신규, step5)
        step2.html          (신규, step5)
        step3.html          (신규, step5)
        step4.html          (신규, step5)
```

#02. 컨트롤러 클래스

1) 클래스 정의

어떤 패키지에 위치하건 상관 없으나 일반적으로 `*.cotrollers` 패키지 안에 모아 놓는다.

컨트롤러는 클래스 정의문 상단에 `@Controller` 어노테이션을 추가해야 한다.

```
import org.springframework.stereotype.Controller;

@Controller
public class HomeController {

}
```

2) 메서드

GET방식의 접근인 경우

`@GetMapping('url경로')` 형식의 어노테이션을 메서드 정의문 상단에 추가한다.

```
package study.spring.params.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {
    @GetMapping("/")
    public String home() {
        return "home";
```

```
    }
}
```

POST 방식의 접근인 경우

`@PostMapping('url경로')` 형식의 어노테이션을 메서드 정의문 상단에 추가한다.

```
package study.spring.params.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class HomeController {
    @PostMapping("/")
    public String home() {
        return "home";
    }
}
```

3) GET, POST 파라미터

View에 변수를 전달하기 위해서는 컨트롤러 메서드 파라미터로 `Model` 객체를 선언한다.

Get, Post 파라미터를 식별하기 위해서는 아래의 형식으로 파라미터를 정의한다.

```
@RequestParam(value="파라미터이름", defaultValue="기본값") 데이터타입 변수명
```

기본 사용 방법

`http://localhost:8080/home?name=lee&kor=98&math=72` 인 경우

```
@GetMapping("/home")
public String post(Model model,
            @RequestParam(value="name", defaultValue="") String nameValue,
            @RequestParam(value="kor", defaultValue=0) int korValue,
            @RequestParam(value="math", defaultValue=0) int mathValue)

    ... 메서드 구현 ...

    return "home";
}
```

파라미터와 변수명을 다르게 지정하는 경우

> 위의 기본 사용 방법에 대한 경우와 동일

```
@RequestParam(value="foo") int fooNumber,
@RequestParam(value="bar") String barString,
@RequestParam(value="ok") boolean isOk
```

파라미터와 변수명이 동일한 경우

파라미터 이름과 변수명이 동일하다면 `@RequestParam`에서 `value`속성을 생략할 수 있다.

```
@RequestParam(defaultValue="") String name,
@RequestParam(defaultValue=0) int kor,
@RequestParam(defaultValue=0) int math
```

전달되지 않은 파라미터에 대한 기본값 설정

```
@RequestParam(defaultValue=0) int foo,
@RequestParam(defaultValue="") String bar,
@RequestParam(defaultValue=true) boolean ok
```

파라미터가 전달되지 않는 것을 허용하는 옵션 추가

```
@RequestParam(required=false) int foo,
@RequestParam(required=false) String bar,
@RequestParam(required=false) boolean ok
```

4) PATH 파라미터

URL의 폴더 형식으로 변수를 포함시키기

```
http://localhost:8080/컨트롤러PATH/{foo}/{bar}
```

`@RequestMapping`의 `value` 속성으로 URL을 지정할 때 중괄호 `{ }` 를 사용하여 변수의 이름을 명시한다.

메서드 파라미터로 @PathVariable 어노테이션을 사용하여 URL에 명시된 변수 이름과 동일한 파라미터를 선언하면 URL에서 해당 위치의 문자열이 메서드 파라미터로 전달된다.

지정된 PATH 파라미터가 전달되지 않을 경우 존재하지 않는 URL로의 접근으로 인식되어 404 에러 (Page Not Found) 가 발생한다.

```
@PathVariable int foo,
@PathVariable String bar
```

URL에 명시된 key와 파라미터 변수 이름을 다르게 지정하는 경우

`@PathVariable(key이름)` 형식으로 어떤 항목에 대한 변수인지 지정해야 한다.

```
http://localhost:8080/컨트롤러PATH/{foo}/{bar}
```

```
@PathVariable("foo") int number,
@PathVariable("bar") String text
```