

# SPRING

## 19강\_Controller 객체 구현 - II

컨트롤러의 URL 매핑과 파라미터 처리 방법에 대해서 학습합니다.

---

19-1 @ModelAttribute

19-2 커맨드 객체 프로퍼티 데이터 타입

19-3 Model & ModelAndView

## 19-1 : @ModelAttribute

@ModelAttribute를 이용하면 커맨드 객체의 이름을 변경할 수 있고,  
이렇게 변경된 이름은 뷰에서 커맨드 객체를 참조할 때 사용된다.

컨트롤러

```
public String memJoin(Member member)
```

```
public String memLogin(Member member) {
```

```
public String memRemove(@ModelAttribute("mem") Member member)
```

뷰

```
ID : ${member.memId}
```

```
ID : ${member.memId}
```

```
ID : ${mem.memId}
```

## 19-2 : 커맨드 객체 프로퍼티 데이터 타입

데이터가 기초데이터 타입인 경우

## memberJoin.html

```
ID : <input type="text" name="memId" >  
PW : <input type="password" name="memPw" >  
MAIL : <input type="text" name="memMail" >  
AGE : <input type="text" name="memAge" size="4" value="0">
```



## Member.java

```
private String memId;  
private String memPw;  
private String memMail;  
private int memAge;
```

## 19-2 : 커맨드 객체 프로퍼티 데이터 타입

데이터가 중첩 커맨드 객체를 이용한 List 구조인 경우

## memberJoin.html

```
PHONE1 : <input type="text"
name="memPhones[0].memPhone1" size="5"> -
<input type="text"
name="memPhones[0].memPhone2" size="5"> -
<input type="text"
name="memPhones[0].memPhone3" size="5">
PHONE2 : <input type="text"
name="memPhones[1].memPhone1" size="5"> -
<input type="text"
name="memPhones[1].memPhone2" size="5"> -
<input type="text"
name="memPhones[1].memPhone3" size="5">
```



## Member.java

```
private List<MemPhone> memPhones;
```

## 19-3 : Model & ModelAndView

컨트롤러에서 뷰에 데이터를 전달하기 위해 사용되는 객체로 Model과 ModelAndView가 있다. 두 객체의 차이점은 Model은 뷰에 데이터만을 전달하기 위한 객체이고, ModelAndView는 데이터와 뷰의 이름을 함께 전달하는 객체이다.

### Model

```
@RequestMapping(value = "/memModify", method = RequestMethod.POST)
public String memModify(Model model, Member member) {

    Member[] members = service.memberModify(member);

    model.addAttribute("memBef", members[0]);
    model.addAttribute("memAft", members[1]);

    return "memModifyOk";
}
```

**memModifyOk.jsp**

ID : \${memBef.memId}  
ID : \${memAft.memId}

## 19-3 : Model & ModelAndView

컨트롤러에서 뷰에 데이터를 전달하기 위해 사용되는 객체로 Model과 ModelAndView가 있다. 두 객체의 차이점은 Model은 뷰에 데이터만을 전달하기 위한 객체이고, ModelAndView는 데이터와 뷰의 이름을 함께 전달하는 객체이다.


### ModelAndView

```
@RequestMapping(value = "/memModify", method = RequestMethod.POST)
public String memModify(Model model, Member member) {

    Member[] members = service.memberModify(member);

    model.addAttribute("memBef", members[0]);
    model.addAttribute("memAft", members[1]);

    return "memModifyOk";
}
```



```
@RequestMapping(value = "/memModify", method = RequestMethod.POST)
public ModelAndView memModify(Member member) {

    Member[] members = service.memberModify(member);

    ModelAndView mav = new ModelAndView();
    mav.addObject("memBef", members[0]);
    mav.addObject("memAft", members[1]);

    mav.setViewName("memModifyOk");

    return mav;
}
```

# 19-3 : Model & ModelAndView

## Model 이용

```
@RequestMapping(value = "/memModify", method = RequestMethod.POST)
public String memModify(Model model, Member member) {

    Member[] members = service.memberModify(member);

    model.addAttribute("memBef", members[0]);
    model.addAttribute("memAft", members[1]);

    return "memModifyOk";
}
```

데이터 이름 & 데이터

뷰이름

## ModelAndView 이용

```
@RequestMapping(value = "/memModify", method = RequestMethod.POST)
public ModelAndView memModify(Member member) {

    Member[] members = service.memberModify(member);

    ModelAndView mav = new ModelAndView();
    mav.addObject("memBef", members[0]);
    mav.addObject("memAft", members[1]);

    mav.setViewName("memModifyOk");

    return mav;
}
```