

# 스프링 MVC 웹 요청 처리

## - 뷰 지정 -

## 뷰 지정

---

### ❖ 뷰 이름 명시적 지정: ModelAndView와 String 리턴 타입

- 뷰 이름 명시적 지정
  - ModelAndView를 리턴
    - ModelAndView : 생성자를 이용한 뷰 이름 지정

```
@RequestMapping("/index")
public ModelAndView index() {
    ModelAndView mav = ModelAndView("index");
    ...
    return mav;
}
```

- setName() 메서드를 이용한 뷰 이름 지정

```
ModelAndView mav = new ModelAndView();
mav.setName("index");
```

## 뷰 지정

---

### ❖ 뷰 이름 명시적 지정: ModelAndView와 String 리턴 타입

- 뷰 이름 명시적 지정
  - String 리턴

```
@RequestMapping("/help/main")  
public String helpMain(ModelMap model) {  
    ...  
    return "index";  
}
```

# 뷰 지정

---

## ❖ 뷰 이름 자동 지정

- RequestToViewNameTranslate를 이용한 뷰 이름 자동 결정
  - URL을 이용해서 결정
  - 적용되는 경우
    - 리턴 타입이 Model이나 Map인 경우
    - 리턴 타입이 void이면서 ServletResponse나 HttpServletResponse 타입의 파라미터가 없는 경우

```
@RequestMapping("/search/game2")
public Map<String, Object> search() {
    HashMap<String, Object> model = new HashMap<String, Object>();
    ...
    return model;
}
```

# 뷰 지정

---

## ❖ 뷰 이름 자동 지정

- RequestToViewNameTranslate의 기본 구현체
  - DefaultRequestToViewNameTranslator 사용
  - 요청 URL로부터 맨 앞의 슬래시와 확장자를 제외한 나머지 부분을 뷰 이름으로 사용
  - /search/game2 → search/game2

# 뷰 지정

---

## ❖ 리다이렉트 뷰

- 뷰 이름 앞에 `redirect:` 접두어 사용
  - `redirect:/bbs/list` → 현재 서블릿 컨텍스트에 대한 상대적인 경로로 리다이렉트
  - `redirect:http://host/bbs/list` → 지정한 절대 URL로 리다이렉트

```
ModelAndView mav = new ModelAndView();  
mav.setViewName("redirect:/error");  
return mav;
```

# 스프링 MVC 웹 요청 처리

## - 모델 생성 -

# 모델 생성하기

---

## ❖ 뷰에 전달되는 모델 데이터

- 컨트롤러에서 뷰로 전달되는 모델의 종류
  - `@RequestMapping` 어노테이션 적용 메서드의 리턴 객체 :
    - `ModelAndView`,
    - `Model`,
    - `Map`
  - 커맨드 객체
  - `@ModelAttribute` 어노테이션 적용 메서드의 리턴 객체
  - 메서드의 `Map`, `Model`, `ModelMap` 타입의 파라미터를 통해 설정된 모델



# 모델 생성하기

---

## ❖ 뷰에 전달되는 모델 데이터

- 컨트롤러에서 뷰로 전달되는 모델의 종류

```
@Controller
public class GameSearchController {
    private SearchService searchService;

    @ModelAttribute("searchTypeList")
    public List<SearchType> referenceSearchTypeList() {
        List<SearchType> options = new ArrayList<SearchType>();
        options.add(new SearchType(1, "전체"));
        options.add(new SearchType(2, "아이템"));
        options.add(new SearchType(3, "캐릭터"));

        return options;
    }
}
```

# 모델 생성하기

## ❖ 뷰에 전달되는 모델 데이터

- 컨트롤러에서 뷰로 전달되는 모델의 종류

```
@RequestMapping("/search/game")
public ModelAndView search(
    @ModelAttribute("command") SearchCommand command,
    ModelMap model) {

    String[] queryList = getPopularQueryList();
    model.addAttribute("popularQueryList", queryList);

    ModelAndView mav = new ModelAndView("search/game");
    SearchResult result = searchService.search(command);
    mav.addObject("searchResult", result);

    return mav;
}
...
```

# 모델 생성하기

## ❖ 뷰에 전달되는 모델 데이터

- 뷰 코드에서의 모델 데이터 사용

```
<body>
인기 키워드:
<c:forEach var="popularQuery" items="${popularQueryList}">
    ${popularQuery}
</c:forEach>

<form action="game">
<select name="type">
    <c:forEach var="searchType" items="${searchTypeList}">
        <option value="${searchType.code}"
            <c:if test="${command.type==searchType.code}">selected</c:if>>
            ${searchType.text}</option>
    </c:forEach>
</select>
<input type="text" name="query" value="${command.query}" />
<input type="submit" value="검색" />
</form>
검색결과: ${searchResult}
</body>
```

# 모델 생성하기

---

## ❖ Map, Model, ModelMap을 통한 모델 설정

- 세 가지 타입 중 한 가지를 파라미터로 전달
  - Map 객체 파라미터

```
@RequestMapping("/search1")
public String search1(Map model) {
    ...
    model.put("result", searchResult);
    ...
}
```

# 모델 생성하기

---

## ❖ Map, Model, ModelMap을 통한 모델 설정

- 세 가지 타입 중 한 가지를 파라미터로 전달
  - Model 객체 파라미터

```
@RequestMapping("/search2")
public String search1(Model model) {
    ...
    model.addAttribute("result", searchResult);
    ...
}
```

- ModelMap 객체 파라미터

```
@RequestMapping("/search3")
public String search1(ModelMap model) {
    model.addAttribute("result", searchResult);
    ...
}
```

# 모델 생성하기

---

## ❖ Map, Model, ModelMap을 통한 모델 설정

- Map과 Model을 리턴
  - Map, Model은 인터페이스
  - Map 리턴 (HashMap 리턴)

```
@RequestMapping("/search1")  
public Map search1() {  
    ...  
    HashMap<String, Object> model = new HashMap<String, Object>();  
    model.put("result", searchResult);  
  
    return model;  
}
```

# 모델 생성하기

---

## ❖ Map, Model, ModelMap을 통한 모델 설정

- Map과 Model을 리턴
  - Model 리턴
    - Model은 인터페이스
    - Model 인터페이스 구현체 : ExtendedModelMap

```
@RequestMapping("/search2")
public Model search1() {
    ...
    Model model = new ExtendedModelMap();
    model.addAttribute("result", searchResult);

    return model;
}
```

# 모델 생성하기

---

## ❖ Map, Model, ModelMap을 통한 모델 설정

### ○ Model 인터페이스의 주요 메서드

- **Model addAttribute(String name, Object value)**
  - value 객체를 name 이름으로 추가. 뷰 코드에서는 name으로 지정한 이름을 통해서 value를 사용
- **Model addAttribute(Object value)**
  - value를 추가. 단순 클래스 이름을 모델 이름으로 사용(첫 글자는 소문자)
  - value가 배열이거나 컬렉션인 경우 첫 번째 원소의 클래스 이름 뒤에 List를 붙여 모델이름으로 사용(첫 글자는 소문자)
- **Model addAllAttributes(Collection<?> values)**
  - addAttribute(Object value) 메서드를 이용해서 컬렉션에 포함된 객체를 차례대로 추가
- **Model addAllAttributes(Map<String, ?> attributes)**
  - Map에 포함된 <키, 값>에 대해 키를 모델 이름으로 사용해서 값을 모델로 추가
- **Model mergeAttributes(Map<String, ?> attributes)**
  - Map에 포함된 <키, 값>을 현재 모델에 추가. 키와 동일한 이름을 갖는 모델 객체가 존재하지 않는 경우에만 추가
- **boolean containsAttribute(String name)**
  - 지정한 이름의 모델 객체를 포함하고 있는 경우 true 리턴



# 모델 생성하기

---

## ❖ Map, Model, ModelMap을 통한 모델 설정

- ModelMap 클래스
  - ModelMap 클래스를 파라미터로 지정하는 경우

```
@RequestMapping("/search/game")
public String search(SearchCommand command, ModelMap model) {
    ...
    model.addAttribute("searchResult", result);
    model.addAttribute("searchTypeList", searchTypeList);
    ...
}
```

# 모델 생성하기

## ❖ ModelAndView를 통해 모델 생성

- ModelAndView
  - 뷰 지정과 뷰에 전달할 값을 저장
  - 뷰의 지정
    - 생성자로 지정
    - `setViewName()` 메서드로 지정
  - 값 저장
    - `addObject(String name, Object value)`

```
@RequestMapping("/search/game")
public ModelAndView search(SearchCommand command) {
    ...
    ModelAndView mav = new ModelAndView();
    mav.setViewName("search/game");
    mav.addObject("searchResult", searchResult);
    mav.addObject("searchTypeList", typeList);
    return mav;
}
```

# 모델 생성하기

## ❖ ModelAndView를 통해 모델 생성

### ○ ModelAndView

- Mapo 객체의 저장 내용을 ModelAndView에 추가
  - addAllObjects(Map modelMap) 메서드 이용

```
Map referenceMap = referenceData();  
mav.addAllObjects(referenceMap);
```

- 생성자를 사용해서 뷰 이름과 Map을 전달

```
Map referenceMap = referenceData();  
return new ModelAndView("search/game", referenceMap);
```

- 뷰에 전달할 객체가 한 개 뿐인경우

```
return new ModelAndView("search/game", "result", searchResult);
```

# 모델 생성하기

---

## ❖ @ModelAttribute 어노테이션 적용 메서드에 전달 가능한 파라미터

- @RequestMapping 적용 메서드와 동일한 파라미터를 가질 수 있음
  - HttpServletRequest, Locale, @RequestParam 적용 파라미터, @PathVariable 적용 파라미터 등

```
@ModelAttribute("command")
public MemberInfo formBacking(HttpServletRequest request) {
    if(request.getMethod().equalsIgnoreCase("GET")) {
        ...
    } else {
        ...
    }
    return new MemberInfo();
}
```

# 모델 생성하기

---

## ❖ @ModelAttribute 어노테이션을 이용한 모델 데이터 처리

- @ModelAttribute 어노테이션 기능
  - @RequestMapping 어노테이션이 적용되지 않은 별도 메서드로 모델에 추가될 객체를 생성
  - 커맨드 객체의 초기화 작업을 수행

# 모델 생성하기

## ❖ 참조 데이터 생성

- @ModelAttribute 어노테이션을 통한 참조 데이터 생성

```
@Controller
public class GameSearchController {
    private SearchService searchService;

    @ModelAttribute("searchTypeList")
    public List<SearchType> referenceSearchTypeList() {
        List<SearchType> options = new ArrayList<SearchType>();

        options.add(new SearchType(1, "전체"));
        options.add(new SearchType(2, "아이템"));
        options.add(new SearchType(3, "캐릭터"));
        return options;
    }

    @ModelAttribute("popularQueryList")
    public String[] getPopularQueryList() {
        return new String[] {"게임", "창천2", "위메이드"};
    }
}
```

# 모델 생성하기

---

## ❖ 참조 데이터 생성

- @ModelAttribute 어노테이션을 통한 참조 데이터 생성

```
@RequestMapping("/search/main")
public String main() {
    return "search/main";
}

@RequestMapping("/search/game")
public ModelAndView search(
    @ModelAttribute("command") SearchCommand command) {
    ModelAndView mav = new ModelAndView("search/game");
    SearchResult result = searchService.search(command);
    mav.addObject("searchResult", result);
    return mav;
}

...
}
```

# 모델 생성하기

---

## ❖ 커맨드 객체 초기화


- @ModelAttribute를 이용한 커맨드 객체 초기화
  - GET 요청시 폼에서 사용할 디폴트 값을 command 객체에 미리 초기화
  - POST 요청시 폼에 입력한 값을 커맨드 객체로 받음



## 모델 생성하기

### ❖ 커맨드 객체 초기화 : @ModelAttribute를 이용한 커맨드 객체 초기화

```
@Controller
@RequestMapping("/account/create")
public class CreateAccountController {
    @ModelAttribute("command")
    public MemberInfo formBacking(HttpServletRequest request) {
        if(request.getMethod().equalsIgnoreCase("GET")) {
            MemberInfo mi = new MemberInfo(디폴트 값 설정);
            return mi;
        } else {
            return new MemberInfo();
        }
    }
    @RequestMapping(method=RequestMethod.GET)
    public String form() {
        return "account/creationForm";
    }
    @RequestMapping(method=RequestMethod.POST) {
    public String submit(@ModelAttribute("command") MemberInfo memberInfo) {
        return "account/created";
    }
    ...
}
```



모델 이름 동일