11

Model 및 Form 데이터 사용하기

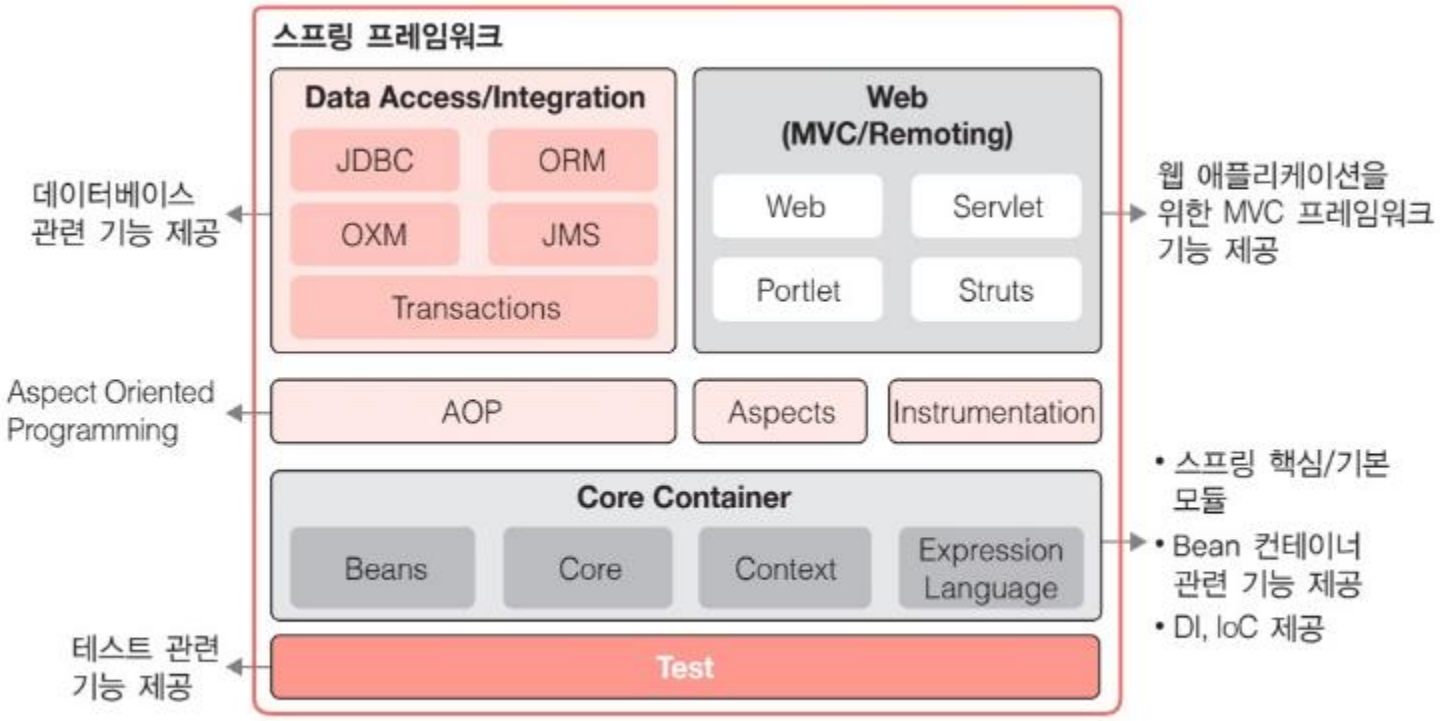
2

Model 사용하기

Spring Framework

Spring 모듈

- Spring은 **다양한 모듈(라이브러리)**로 구성
- 개발자는 프로젝트 특성에 맞는 모듈을 찾아 적절히 사용
- 모듈을 사용하려면 필요한 모듈에 대한 '**의존(dependency) 설정**' 추가

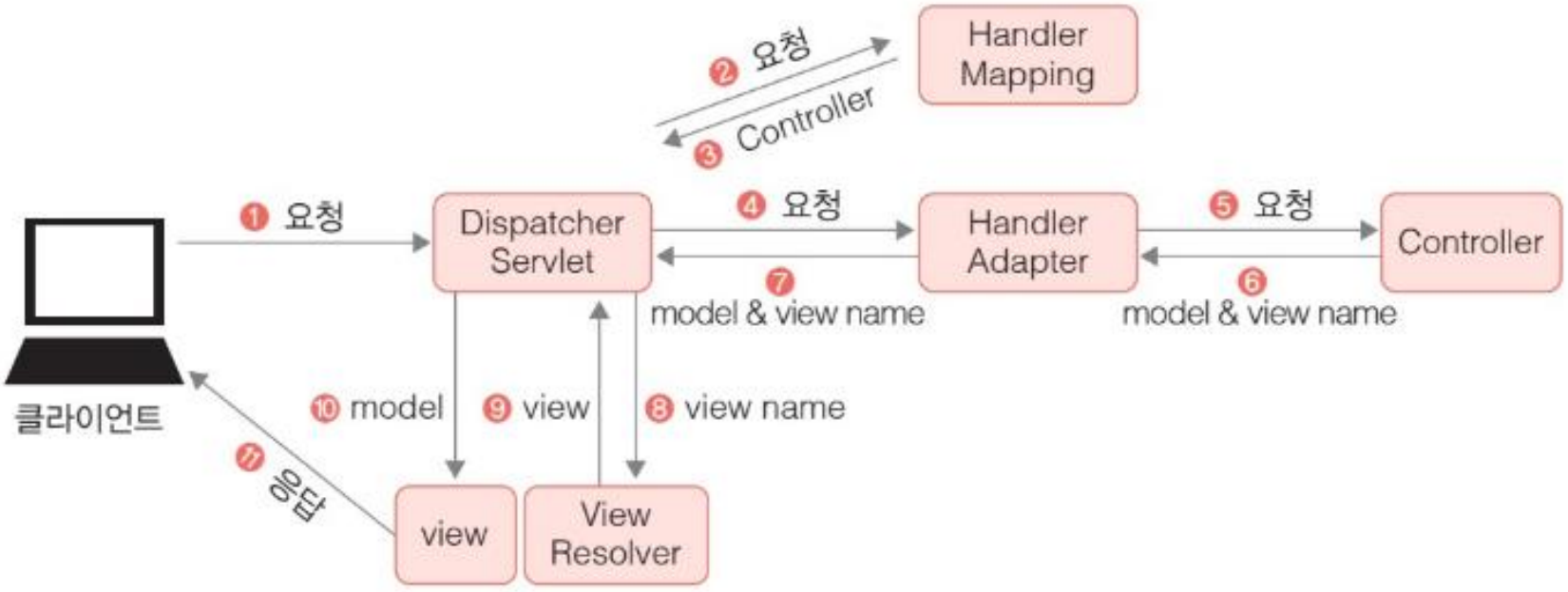


필요에 따라 특정 모듈만 이용할 수 있는 경량 컨테이너인 스프링

Spring MVC의 개념

- **기본 개념**
  - **웹 애플리케이션**을 **개발**하기 위한 **전용 프레임워크**
  - 기본적으로는 **MVC 패턴**을 따름

- **Spring MVC의 흐름**



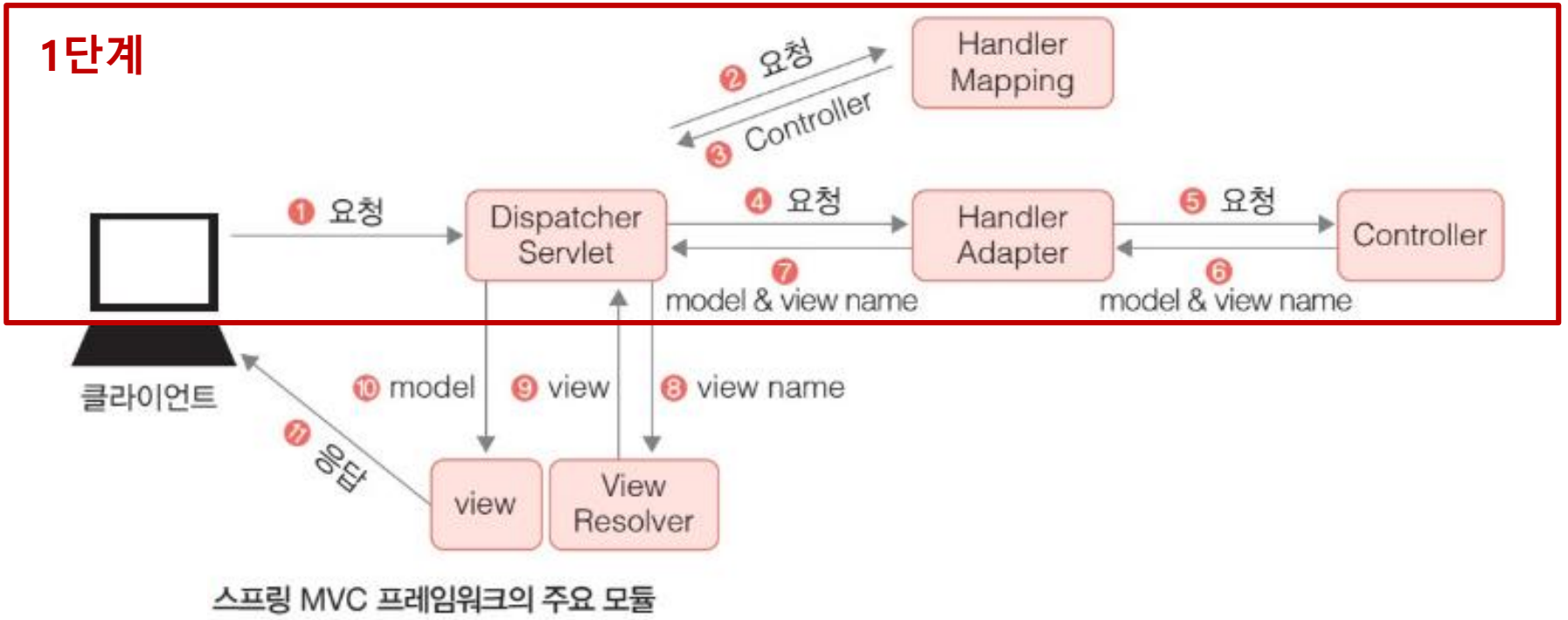
스프링 MVC 프레임워크의 주요 모듈

Spring MVC의 개념

기본 개념

- 웹 애플리케이션을 개발하기 위한 전용 프레임워크
- 기본적으로는 MVC 패턴을 따름

Spring MVC의 흐름

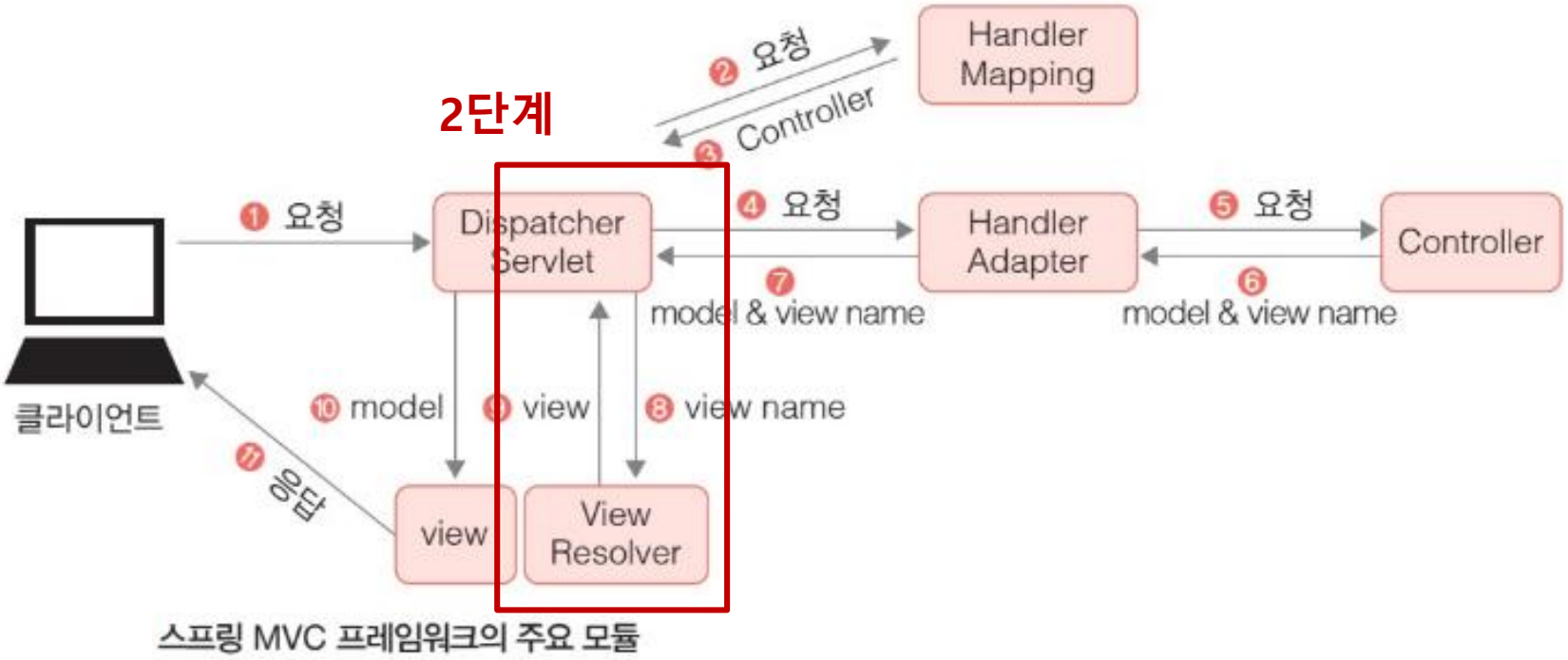


스프링 MVC 프레임워크의 주요 모듈

Spring MVC의 개념

- **기본 개념**
  - 웹 애플리케이션을 개발하기 위한 전용 프레임워크
  - 기본적으로는 MVC 패턴을 따름

- **Spring MVC의 흐름**

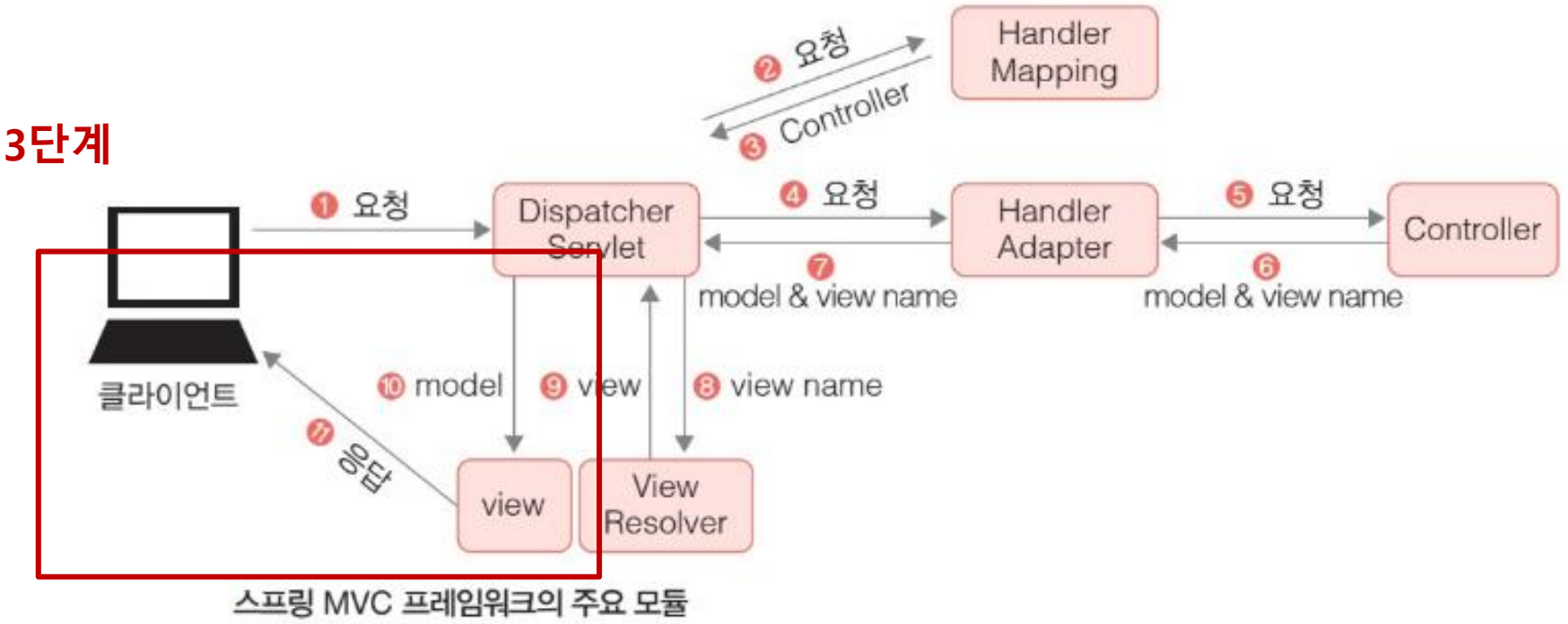


스프링 MVC 프레임워크의 주요 모듈

Spring MVC의 개념

- **기본 개념**
  - 웹 애플리케이션을 개발하기 위한 전용 프레임워크
  - 기본적으로는 MVC 패턴을 따름

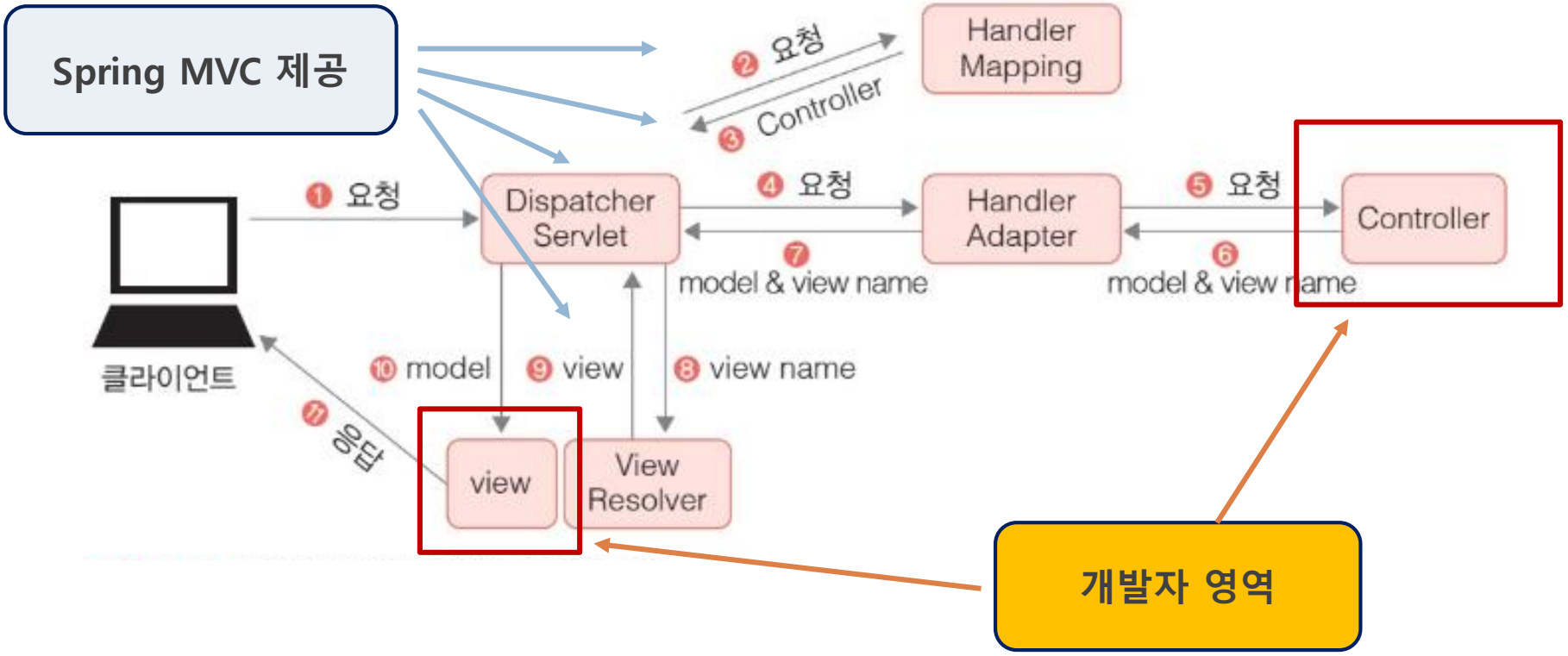
- **Spring MVC의 흐름**



스프링 MVC 프레임워크의 주요 모듈

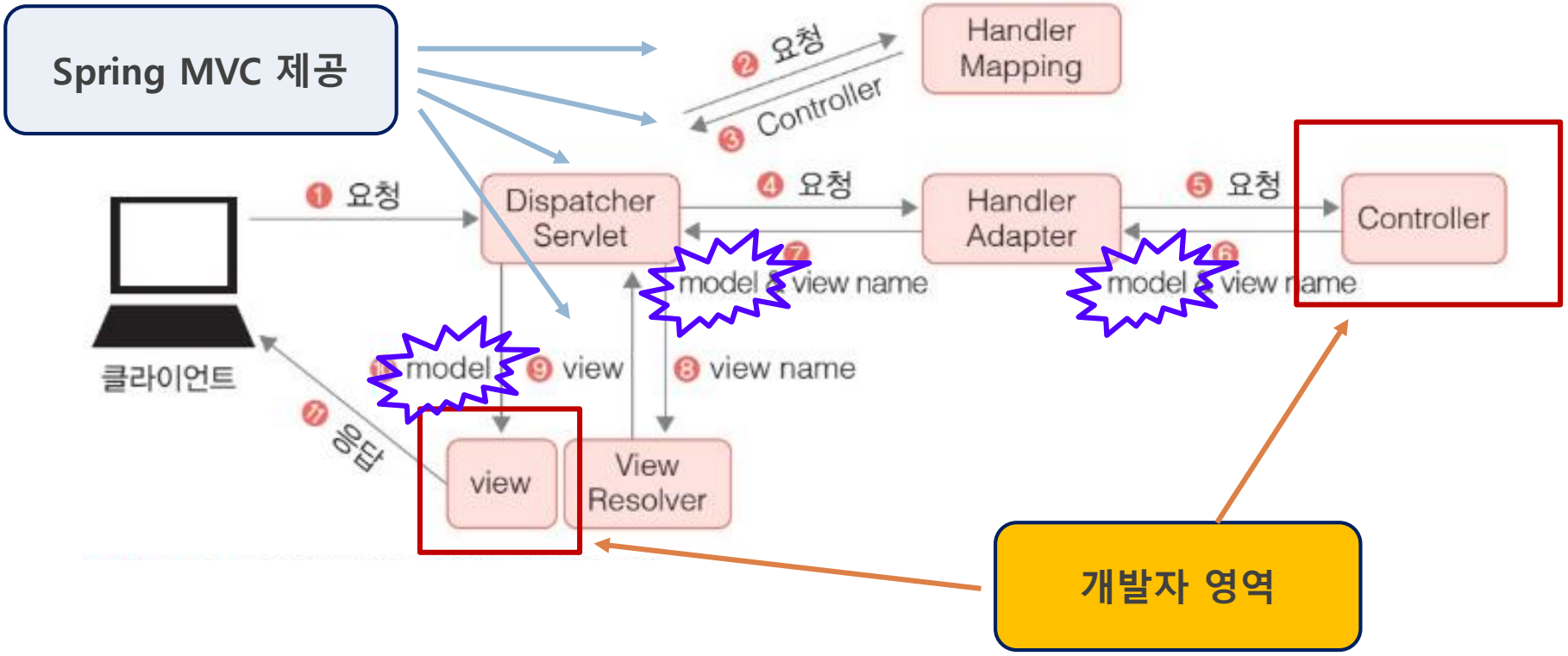
Spring MVC 프레임워크를 사용하는 이유

- **Spring MVC 프레임워크를 사용하는 이유**



Spring MVC 프레임워크를 사용하는 이유

- **Spring MVC 프레임워크를 사용하는 이유**



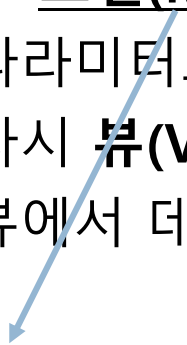
모델(Model) 사용하기

- **컨트롤러(@Controller)**의 역할
  - **@RequestMapping** 어노테이션이 적용된 메서드에서 파라미터로 **모델(Model), 커맨드 객체 등을 받음**
  - 파라미터로 받은 객체에 **데이터를 저장**
  - 다시 **뷰(View)에 전달**
  - **뷰에서 데이터를 사용**할 수 있게 함

```
 7  @Controller
 8  public class MyController {
 9      @RequestMapping("/")          // localhost:8081/ 호출 시 실행
10      @ResponseBody                 // 리턴값을 직접 화면에 출력
11      public String root() throws Exception {
12          return "JSP in Gradle";
13      }
14
15      @RequestMapping("/test1")     // localhost:8081/test1 호출 시 실행
16      public String test1() {
17          return "test1";           // 실제 호출 될 /WEB-INF/views/test1.jsp
18      }
```

모델 사용하기

- **컨트롤러(@Controller)**의 역할
  - **@RequestMapping** 어노테이션이 적용된 메서드에서 파라미터로 **모델(Model), 커맨드 객체 등을 받음**
  - 파라미터로 받은 객체에 **데이터를 저장**
  - 다시 **뷰(View)에 전달**
  - 뷰에서 데이터를 사용할 수 있게 함



- **모델(Model)**
  - 스프링을 사용한 **웹 애플리케이션 개발**에서 **가장 기본**이 되는 부분

스프링의 모델 사용하기

- JSP 사용을 위한 프로젝트 기본 설정
  - File > New > Spring Starter Project >
    - **week11_01ModelAndCommand**
  - **프로젝트 생성 후 JSP 환경 설정**
  - **build.gradle**에 **디펜던시 추가**
  - **Gradle > Refresh Gradle Project**
  - 프로젝트 > properties > Project Facets
    - **Dynamic Web Module > 5.0**
  - **resources** 폴더
    - **application.properties** 수정
      - **spring.mvc.view.prefix=/WEB-INF/views/**
      - **spring.mvc.view.surfix=.jsp**
    - **JSP 폴더 생성**
      - src > main > **webapp > WEB-INF > views**
    - jsp 파일 생성
      - **views 폴더에 생성**

스프링의 모델 사용하기

- JSP 사용을 위한 프로젝트 기본 설정
  - build.gradle > JSP 사용을 위한 의존성 추가

build.gradle

```
16
17 repositories {
18     mavenCentral()
19 }
20
21 dependencies {
22     implementation 'org.springframework.boot:spring-boot-starter-web'
23     providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
24     testImplementation 'org.springframework.boot:spring-boot-starter-test'
25     testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
26     implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
27     implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api'
28     implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl'
29
30 }
31
32 tasks.named('test') {
33     useJUnitPlatform()
34 }
```

스프링의 모델 사용하기

- JSP 사용을 위한 프로젝트 기본 설정
  - build.gradle > JSP 사용을 위한 의존성 추가

build.gradle

```
16
17 repositories {
18     mavenCentral()
19 }
20
21 dependencies {
22     implementation 'org.springframework.boot:spring-boot-starter-web'
23     providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
24     testImplementation 'org.springframework.boot:spring-boot-starter-test'
25     testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
26     implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
27     implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api'
28     implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl'
29
30 }
31
32 tasks.named('test') {
33     useJUnitPlatform()
34 }
```

프로젝트명 > Gradle > Refresh Gradle Project

프로젝트명 > Properties > Project Facets > Dynamic Web Module 버전 수정 > 5.0

스프링의 모델 사용하기

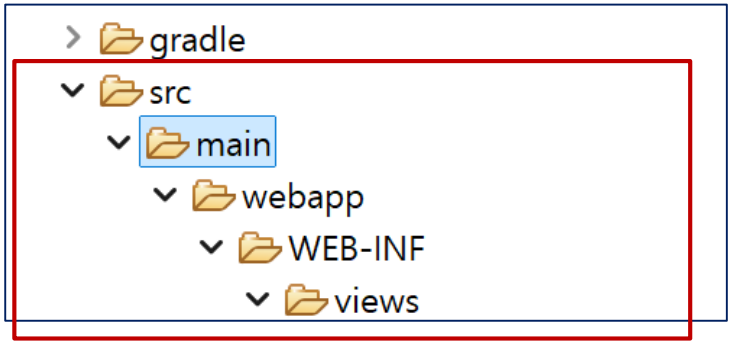
- JSP 사용을 위한 프로젝트 기본 설정
  - application.properties 설정

application.properties

```
1
2
3
4 spring.mvc.view.prefix=/WEB-INF/views/
5 spring.mvc.view.suffix=.jsp
6
```

스프링의 모델 사용하기

- JSP 사용을 위한 폴더 생성
  - 스프링 부트에서 기본으로 제공하는 다른 템플릿 뷰와는 달리 JSP는 src/main/resources의 템플릿 폴더를 사용할 수 없다.
  - JSP를 위해 필요한 폴더는 직접 만들고 지정



스프링의 모델 사용하기

- 리퀘스트 매핑
  - MyController

MyController.java

```java
@Controller // 브라우저를 통한 요청에 대한 처리 담당
public class MyController {
    // 요청 : @RequestMapping()에서 정의
    // 처리 : 메소드 형식으로 정의

    @RequestMapping("/")
    @ResponseBody
    public String root() {
        return "Model & View";
    }

    @GetMapping("/test1")
    public String test1(Model model) {
        //Model 객체를 이용해서 view로 Data 전달
        //데이터만 설정이 가능
        model.addAttribute("name", "홍길동");
        return "test1";
    }
```

스프링의 모델 사용하기

- 리퀘스트 매핑
  - MyController

MyController.java

```java
@Controller // 브라우저를 통한 요청에 대한 처리 담당
public class MyController {
    // 요청 : @RequestMapping()에서 정의
    // 처리 : 메소드 형식으로 정의

    @RequestMapping("/")
    @ResponseBody
    public String root() {
        return "Model & View";
    }

    @GetMapping("/test1")
    public String test1(Model model) {
        //Model 객체를 이용해서 view로 Data 전달
        //데이터만 설정이 가능
        model.addAttribute("name", "홍길동");
        return "test1";
    }
```

브라우저를 통한 요청 형태 정의

스프링의 모델 사용하기

- 리퀘스트 매핑
  - MyController

MyController.java

```
13 @Controller // 브라우저를 통한 요청에 대한 처리 담당
14 public class MyController {
15     // 요청 : @RequestMapping()에서 정의
16     // 처리 : 메소드 형식으로 정의
17
18     @RequestMapping("/")
19     @ResponseBody
20     public String root() {
21         return "Model & View";
22     }
23
24     @GetMapping("/test1")
25     public String test1(Model model) {
26         //Model 객체를 이용해서 view로 Data 전달
27         //데이터만 설정이 가능
28         model.addAttribute("name", "홍길동");
29         return "test1";
30     }
```

브라우저를 통한 요청 형태 정의

요청에 대한 처리를 위한 메서드

스프링의 모델 사용하기

- 리퀘스트 매핑
  - MyController

MyController.java

```
13  @Controller // 브라우저를 통한 요청에 대한 처리 담당
14  public class MyController {
15      // 요청 : @RequestMapping()에서 정의
16      // 처리 : 메소드 형식으로 정의
17
18      @RequestMapping("/")
19      @ResponseBody
20      public String root() {
21          return "Model & View";
22      }
23
24      @GetMapping("/test1")
25      public String test1(Model model) {
26          //Model 객체를 이용해서 view로 Data 전달
27          //데이터만 설정이 가능
28          model.addAttribute("name", "홍길동");
29          return "test1";
30      }
```

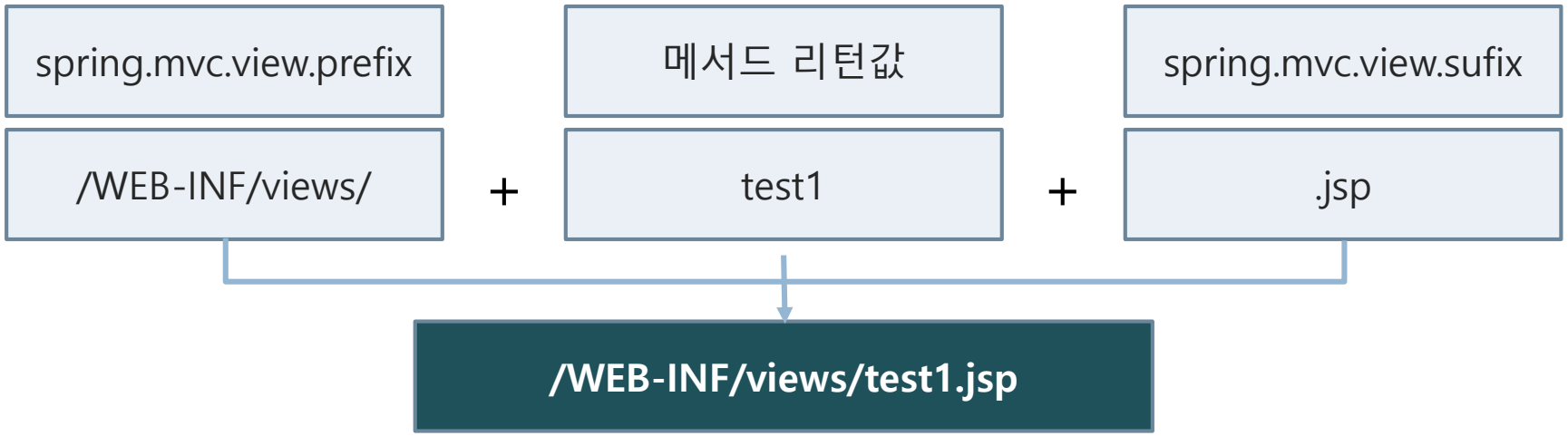
뷰 리졸버(View Resolver)에 의해
/WEB-INF/views/test1.jsp를 찾음

스프링의 모델 사용하기

- JSP 사용을 위한 프로젝트 기본 설정
  - application.properties 설정

application.properties

```
1
2
3
4 spring.mvc.view.prefix=/WEB-INF/views/
5 spring.mvc.view.suffix=.jsp
6
```



스프링의 모델 사용하기

- 리퀘스트 매핑
  - MyController

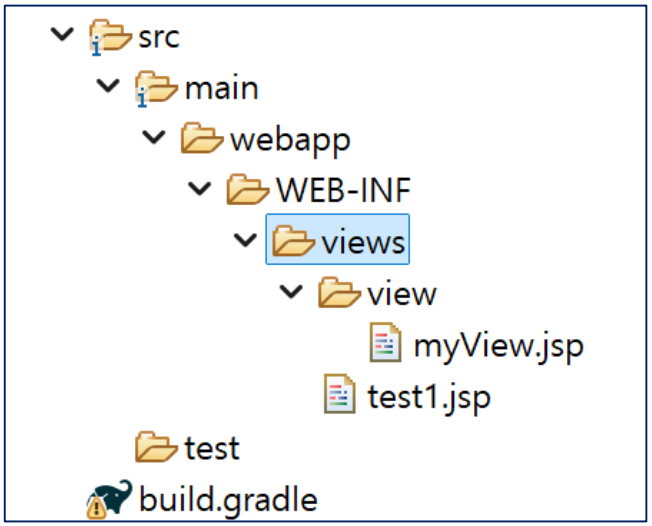
MyController.java

```
13 @Controller // 브라우저를 통한 요청에 대한 처리 담당
14 public class MyController {
15     // 요청 : @RequestMapping()에서 정의
16     // 처리 : 메소드 형식으로 정의
17
18     @RequestMapping("/")
19     @ResponseBody
20     public String root() {
21         return "Model & View";
22     }
23
24     @GetMapping("/test1")
25     public String test1(Model model) {
26         //Model 객체를 이용해서 view로 Data 전달
27         //데이터만 설정이 가능
28         model.addAttribute("name", "홍길동");
29         return "test1";
30     }
```

test1.jsp 에서 **model 객체**에 저장된 **데이터를 사용**하여 test1 뷰 페이지에 **데이터를 출력**할 수 있다.

스프링의 모델 사용하기

- 뷰 만들기
  - test1.jsp

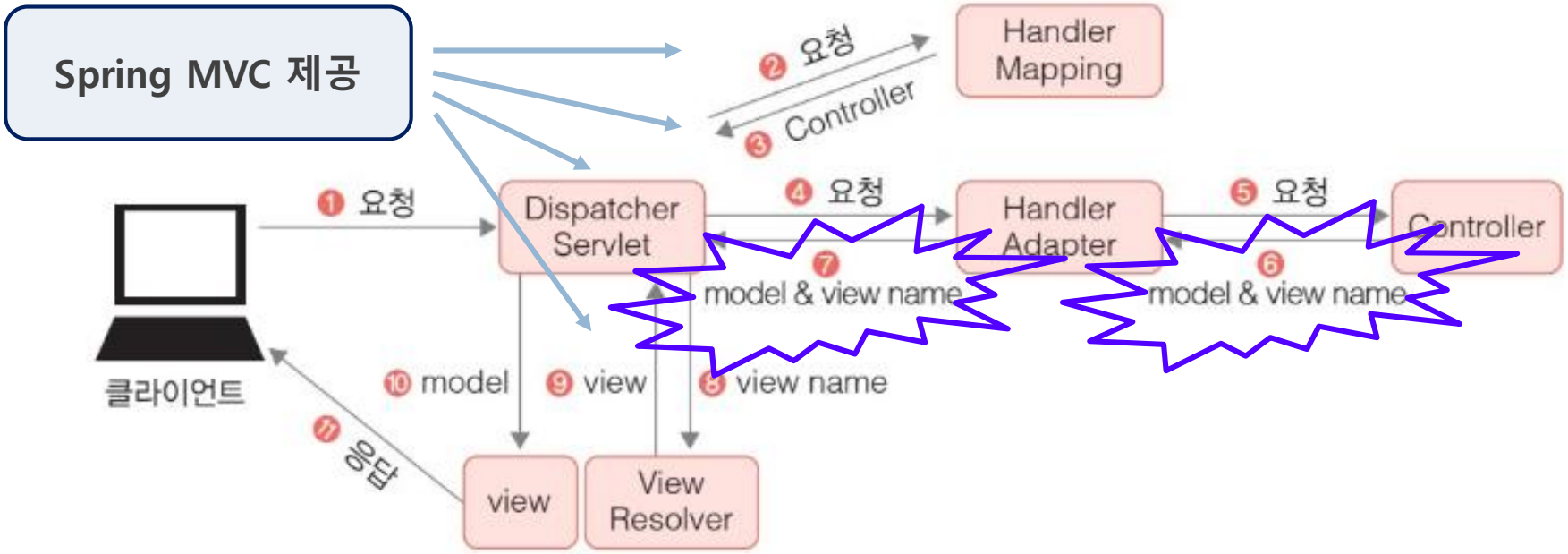


test1.jsp

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>Insert title here</title>
8  </head>
9  <body>
10 <%
11     out.println("<h1>Model : Hello World</h1>");
12 %>
13
14 <br>
15     <h3>당신의 이름은 ${name }입니다. </h3>
16 </body>
17 </html>
```

Spring MVC 프레임워크를 사용하는 이유

- **Spring MVC 프레임워크를 사용하는 이유**



스프링의 모델 사용하기

- 리퀘스트 매핑
  - MyController

```
31
32      @GetMapping("/mv")
33      public ModelAndView test2() {
34          //데이터와 뷰를 동시에 설정이 가능
35          ModelAndView mv = new ModelAndView();
36
37          List<String> list = new ArrayList<>();
38          list.add("test1");
39          list.add("test2");
40          list.add("test3");
41
42          mv.addObject("lists", list);
43          mv.addObject("ObjectTest", "테스트입니다.");
44          mv.addObject("name", "홍길동");
45          mv.setViewName("view/myView");
46
47          return mv;
48      }
49  }
```

스프링의 모델 사용하기

- 리퀘스트 매핑
  - MyController

```
31
32      @GetMapping("/mv")
33      public ModelAndView test2() {
34          //데이터와 뷰를 동시에 설정이 가능
35          ModelAndView mv = new ModelAndView();
36
37          List<String> list = new ArrayList<>();
38          list.add("test1");
39          list.add("test2");
40          list.add("test3");
41
42          mv.addObject("lists", list);
43          mv.addObject("ObjectTest", "테스트입니다.");
44          mv.addObject("name", "홍길동");
45          mv.setViewName("view/myView");
46
47          return mv;
48      }
49  }
```

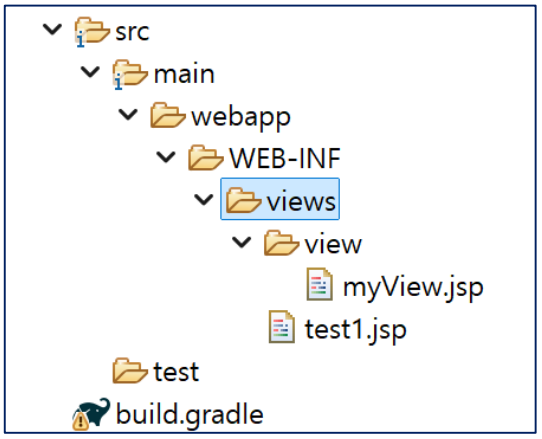
ModelAndView 객체를 리턴

데이터 정보 추가

뷰 정보 추가

스프링의 모델 사용하기

- 뷰 만들기
  - myView.jsp



myView.jsp

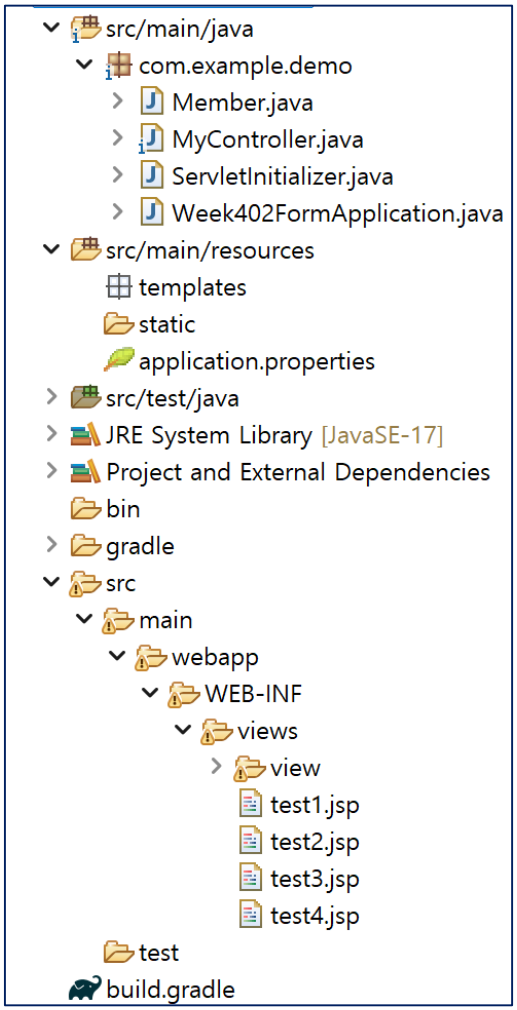
```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
4  <!DOCTYPE html>
5  <html>
6  <head>
7  <meta charset="UTF-8">
8  <title>Insert title here</title>
9  </head>
10 <body>
11 <%
12     out.println("<h1>Model(Sub) : Hello World  </h1>");
13 %>
14 <br>
15 ${ObjectTest }
16 <br>
17 ${lists }
18 <br><br>
19 <c:forEach var="mylist" items="${lists }">
20     ${mylist } <br>
21 </c:forEach>
22 <br>
23 당신의 이름은 ${name }입니다.
24 </body>
25 </html>
```

커맨드 객체를 이용한 데이터 전달

- **커맨드 객체**의 이해
  - **뷰 페이지**의 **폼(Form)에서 입력한 데이터**를 **컨트롤러를 통해 전달받아 처리하는 방법** 학습
  - 스프링은 **커맨드(Command)** 객체를 지원

커맨드 객체를 이용한 데이터 전달

- 커맨드 객체 사용을 위한 클래스 선언

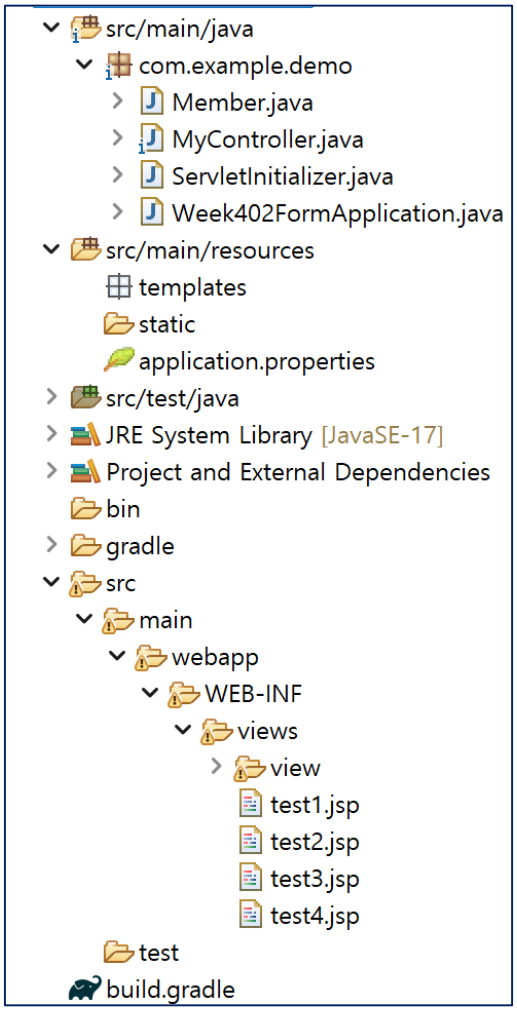


Member.java

```
1
2
3  public class Member {
4      private String id;
5      private String name;
6
7      public String getId() {
8          return id;
9      }
10     public void setId(String id) {
11         this.id = id;
12     }
13     public String getName() {
14         return name;
15     }
16     public void setName(String name) {
17         this.name = name;
18     }
19 }
```

커맨드 객체를 이용한 데이터 전달

- 커맨드 객체 사용을 위한 클래스 선언



커맨드(Command) 객체
== DTO 객체(DB 테이블)와 유사

Member.java

```java

public class Member {
    private String id;
    private String name;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

커맨드 객체를 이용한 데이터 전달

- 리퀘스트 매핑(MyController)

MyController.java

```
13  @Controller
14  public class MyController {
15      @GetMapping("/")
16      public @ResponseBody String root() {
17          return "Form 데이터 전달받아 사용하기";
18      }
19
20      @GetMapping(“/test3”)
21      public String test3 (HttpServletRequest request, Model model) {
22          String id = request.getParameter("id");
23          String name = request.getParameter("name");
24
25          model.addAttribute("id", id);
26          model.addAttribute("name", name);
27          return “test3”;
28      }
```

request 객체로부터 입력된 파라미터를 추출하는 JSP의 전형적인 방법

커맨드 객체를 이용한 데이터 전달

- 뷰 페이지
  - test3.jsp

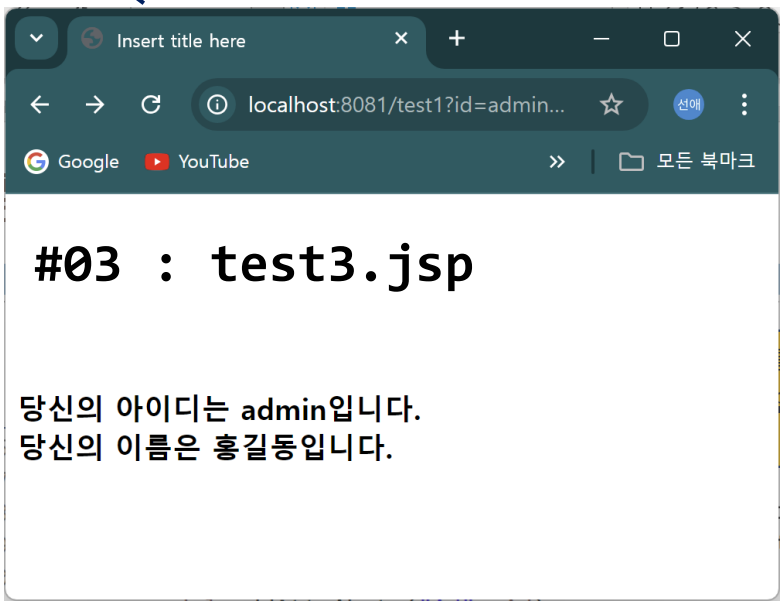
```
 9 <body>
10  <h1>#03 : test3.jsp </h1>
11 <br>
12 <h3>
13     당신의 아이디는 ${id }이고,<br>
14     당신의 이름은 ${name }입니다.
15 </h3>
16 </body>
```

커맨드 객체를 이용한 데이터 전달

- 뷰 페이지
  - test3.jsp

```
 9 <body>
10  <h1>#03 : test3.jsp </h1>
11 <br>
12 <h3>
13     당신의 아이디는 ${id }이고,<br>
14     당신의 이름은 ${name }입니다.
15 </h3>
16 </body>
```

localhost:8081/test3?id=admin&name=홍길동



커맨드 객체를 이용한 데이터 전달

- 리퀘스트 매핑(MyController)

@RequestParam 어노테이션을 이용하여
매개변수에 직접 파라미터 값을 대입

```
29
30      @GetMapping(“/test4”)
31      public String test4  (@RequestParam("id") String id,
32                            @RequestParam("name") String name,
33                            Model model) {
34          // 폼으로 입력되는 파라미터가 많아지면 매개변수가 복잡해진다.
35          model.addAttribute("id", id);
36          model.addAttribute("name", name);
37          return “test4”;
38      }
39
40
41
42
43
44
45
```

커맨드 객체를 이용한 데이터 전달

- 뷰 페이지들

test2.jsp

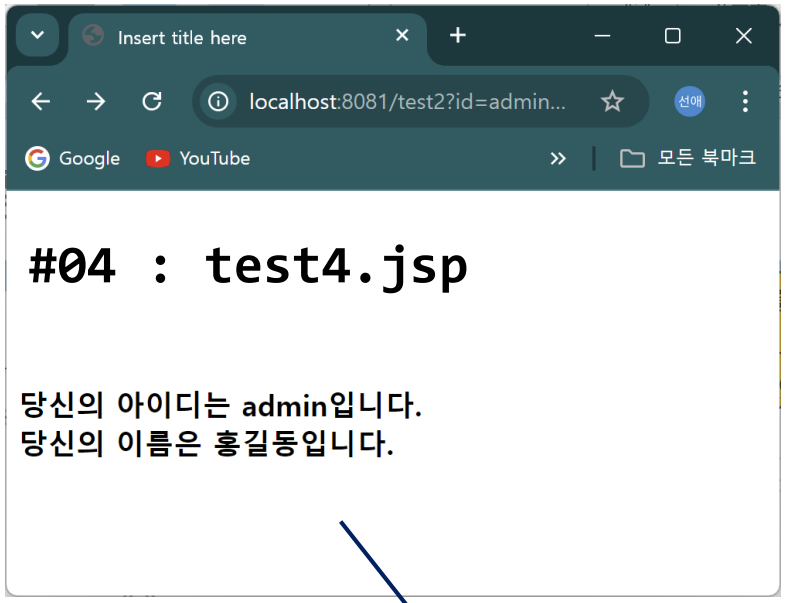
```
9  <body>
10
11 <h1>#04 : test4.jsp </h1>
12
13 <br>
14 <h3>
15     당신의 아이디는 ${id }이고,<br>
16     당신의 이름은 ${name }입니다.
17 </h3>
18 </body>
```

커맨드 객체를 이용한 데이터 전달

- 뷰 페이지들

test2.jsp

```
9  <body>
10
11 <h1>#04 : test4.jsp </h1>
12
13 <br>
14 <h3>
15     당신의 아이디는 ${id }이고,<br>
16     당신의 이름은 ${name }입니다.
17 </h3>
18 </body>
```



localhost:8081/test4?id=admin&name=홍길동

커맨드 객체를 이용한 데이터 전달

- 리퀘스트 매핑(MyController)

path 자체에 변수를 넣어서 호출할 수도 있다.
이 때, 변수인지를 알려주기 위해
@PathVariable 어노테이션을 사용한다.

```
46
47      @GetMapping("/test5/{id}/{name}")
48      public String       test5 (@PathVariable("id") String id,
49                                 @PathVariable("name") String name,
50                                 Model model) {
51          model.addAttribute("id", id);
52          model.addAttribute("name", name);
53          return "test5";
54      }
```

커맨드 객체를 이용한 데이터 전달

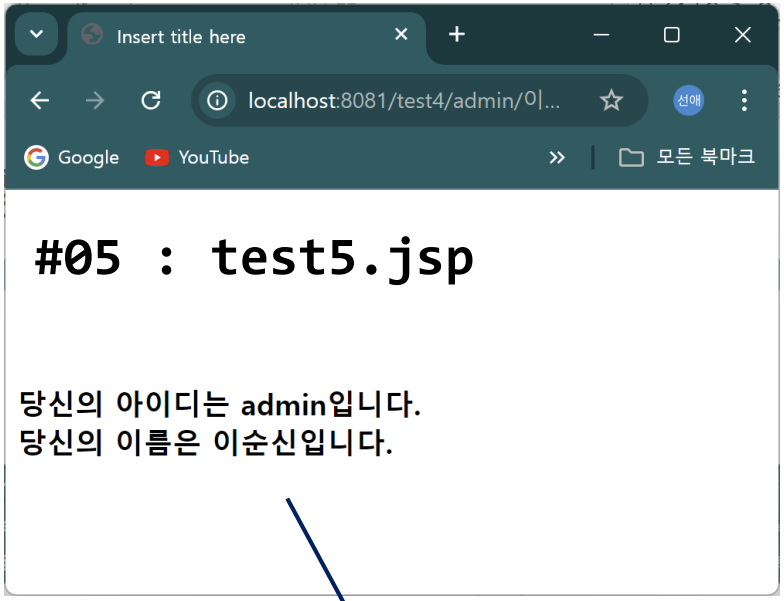
- 뷰 페이지들

```
 9 <body>
<h1>#05 : test5.jsp </h1>
11
12 <br>
13 <h3>
14     당신의 아이디는 ${id }이고,<br>
15     당신의 이름은 ${name }입니다.
16 </h3>
17 </body>
```

커맨드 객체를 이용한 데이터 전달

- 뷰 페이지들

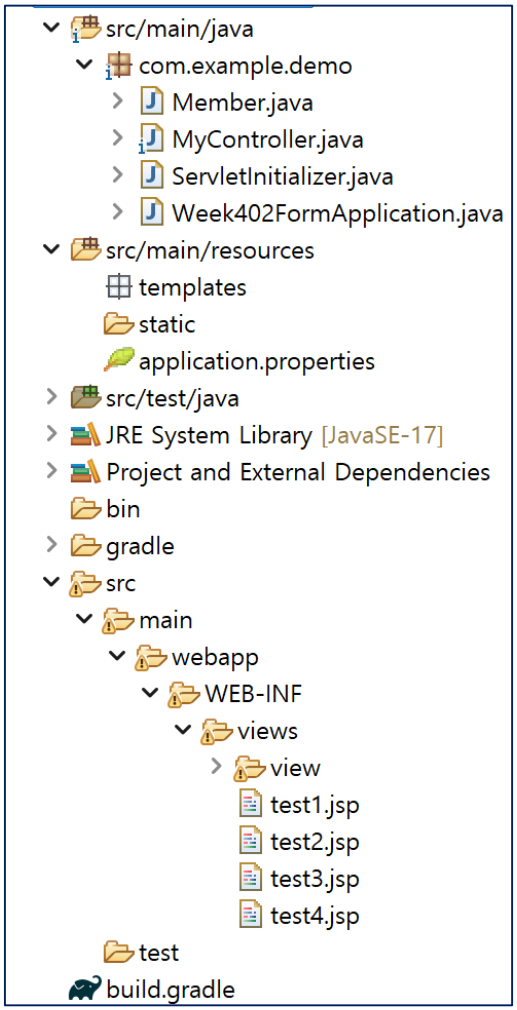
```
9  <body>
<h1>#05 : test5.jsp </h1>
11
12 <br>
13 <h3>
14      당신의 아이디는 ${id }이고,<br>
15      당신의 이름은 ${name }입니다.
16 </h3>
17 </body>
```



localhost:8081/test5/admin/이순신

커맨드 객체를 이용한 데이터 전달

- 커맨드 객체 사용을 위한 클래스 선언



커맨드(Command) 객체
== DTO 객체(DB 테이블)와 유사

Member.java

```
1
2
3  public class Member {
4      private String id;
5      private String name;
6
7      public String getId() {
8          return id;
9      }
10     public void setId(String id) {
11         this.id = id;
12     }
13     public String getName() {
14         return name;
15     }
16     public void setName(String name) {
17         this.name = name;
18     }
19 }
```

커맨드 객체를 이용한 데이터 전달

- 리퀘스트 매핑(MyController)

```
39
40      @GetMapping("/test6")
41      public String test6 (Member member )                    {
42          // 파라미터와 일치하는 빈을 만들어서 사용할 수도 있다.
43          // View 페이지에서 model을 사용하지 않고 member를 사용한다.
44          return "test6";
45      }
```

폼 파라미터와 이름이 같은 변수를 가진 **커맨드 객체를 이용**하면 쉽고 간편하게 많은 데이터를 받아서 처리할 수 있다.

member 커맨드 객체도 뷰에 같이 전달된다.

커맨드 객체를 이용한 데이터 전달

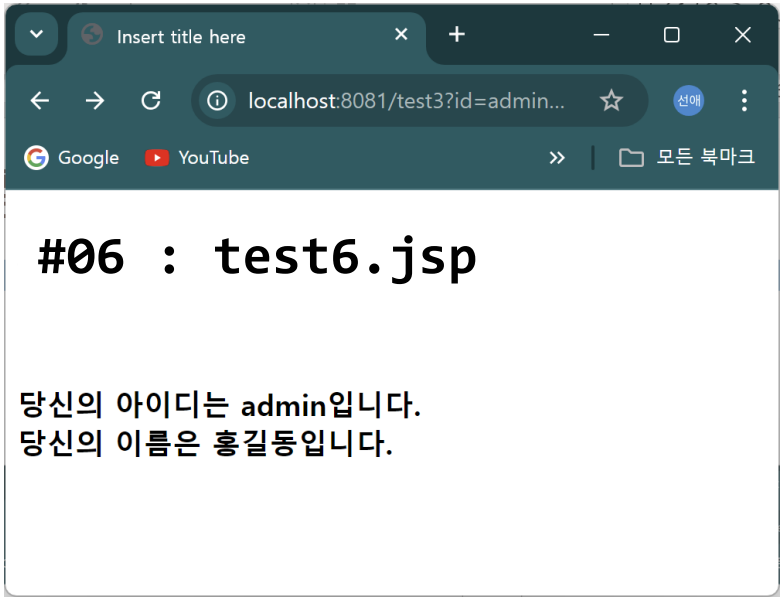
- 뷰 페이지들

```
9  <body>
10  <h1>#06 : test6.jsp </h1>
11
12 <br>
13 <h3>
14      당신의 아이디는 ${member.id }이고,<br>
15      당신의 이름은 ${member.name }입니다.
16 </h3>
17 </body>
```

커맨드 객체를 이용한 데이터 전달

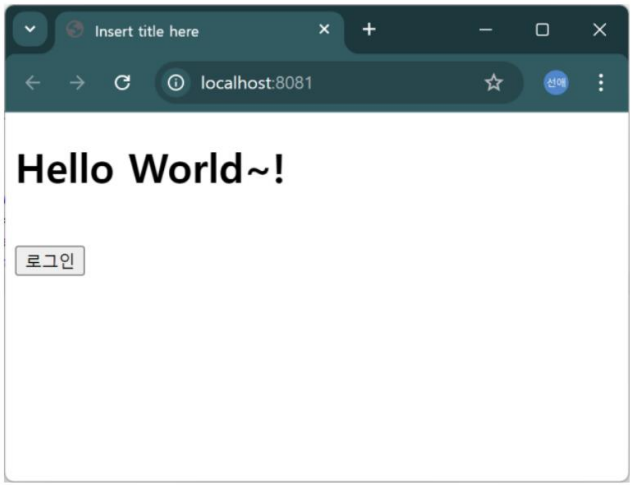
- URL 에서 직접 요청 테스트

localhost:8081/test6?id=admin&name=홍길동

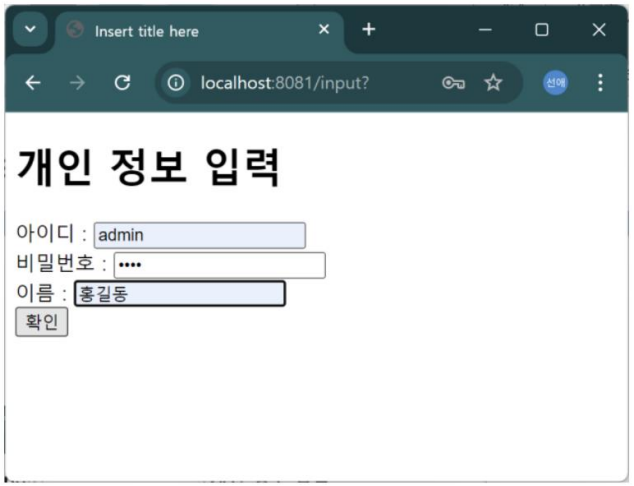


실습(week11_02FormAndModel)

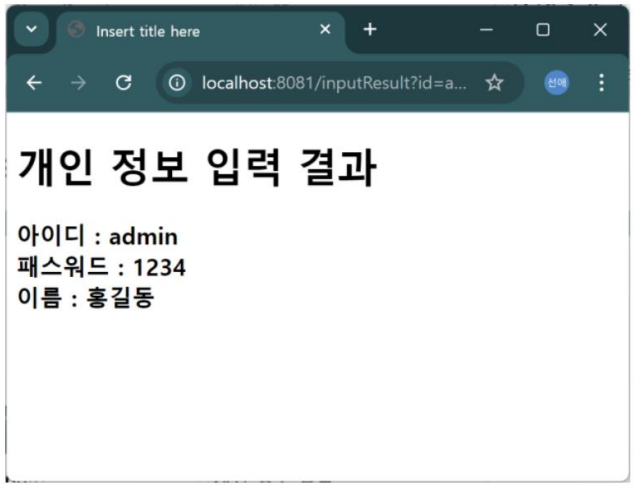
- **입력 폼** 사용
- **@RequestParam** 어노테이션 사용
- **Model 객체**에 **데이터 저장** 후 **뷰에 전달**



index.jsp



inputForm.jsp



inputResult.jsp

실습(week11_02FormAndModel)

- MyController

MyController.java

```
@Controller
public class MyController {
    @RequestMapping("/")
    public String root() {
        return "index"; }
}
```

"/" 요청이 들어오면

/WEB-INF/views/index.jsp 뷰 페이지 실행

실습(week11_02FormAndModel)

- index.jsp

index.jsp

index.jsp

```
1  <%@ page language="java" contentType="text/htm
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>Insert title here</title>
8  </head>
9  <body>
10     <form action="/Login">
11         <h1>Hello World~!</h1>
12         <br>
13         <input type="submit" value="로그인">
14     </form>
15 </body>
16 </html>
```

실습(week11_02FormAndModel)

- index.jsp

index.jsp

index.jsp

```
1  <%@ page language="java" contentType="text/htm
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>Insert title here</title>
8  </head>
9  <body>
10     <form action="/Login">
11         <h1>Hello World~!</h1>
12         <br>
13         <input type="submit" value="로그인">
14     </form>
15 </body>
16 </html>
```

로그인 submit 동작이 실행되면
"/login" 요청이 발생

실습(week11_02FormAndModel)

- MyController

MyController.java

```
@Controller
public class MyController {
    @RequestMapping("/")
    public String root() {
        return "index"; }

    @RequestMapping("/login")
    public String login() {
        return "loginForm"; }
}
```

"/login" 요청이 들어오면

/WEB-INF/views/loginForm.jsp 뷰 페이지 실행

실습(week11_02FormAndModel)

- index.jsp / loginForm.jsp

index.jsp

index.jsp

```
1  <%@ page language="java" contentType="text/htm
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>Insert title here</title>
8  </head>
9  <body>
10     <form action="/login">
15 </body>
16 </html>
```

개인 정보 입력 submit 동작이 실행되면 "/inputResult" 요청이 발생

loginForm.jsp

```
1  <%@ page language="java" contentType="text/html; charset
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>Insert title here</title>
8  </head>
9  <body>
10     <h1>개인 정보 입력</h1>
11     <form action="/inputResult">
12         아이디 : <input type="text" name="id"> <br>
13         비밀번호 : <input type="password" name="pw"> <br>
14         이름 : <input type="text" name="name"> <br>
15         <input type="submit" value="확인"> <br>
16     </form>
17 </body>
18 </html>
```

실습(week11_02FormAndModel)

- MyController

MyController.java

```
8  @Controller
9  public class MyController {
0      @RequestMapping("/")
1      public String root() {
2          return "index"; }
3
4      @RequestMapping("/login")
5      public String login() {
9
0
1
2      @RequestMapping("/inputResult")
3      public String inputResult(@RequestParam("id") String id,
4                                @RequestParam("pw") String pw,
5                                @RequestParam("name") String name,
6                                Model model) {
7          model.addAttribute("id", id);
8          model.addAttribute("pw", pw);
9          model.addAttribute("name", name);
0          return "inputResult";    }
1  }
```

"/inputResult" 요청이 들어오면

입력된 데이터를 @RequestParam()을 이용해서 받고, Model 객체에 저장해서 뷰 페이지로 전달

/WEB-INF/views/inputResult.jsp 뷰 페이지 실행

실습(week11_02FormAndModel)

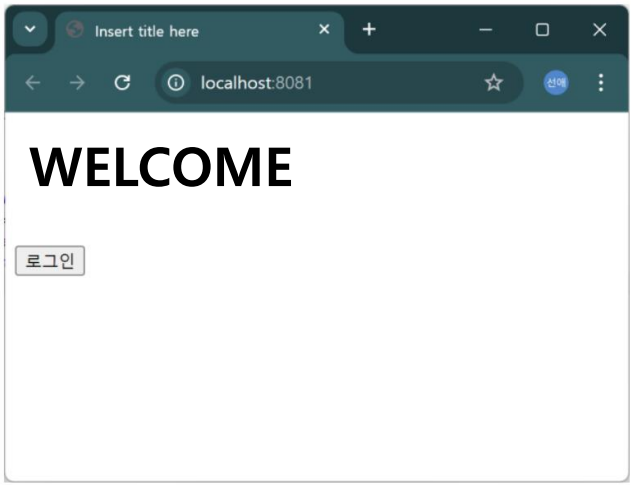
- inputResult.jsp

inputResult.jsp

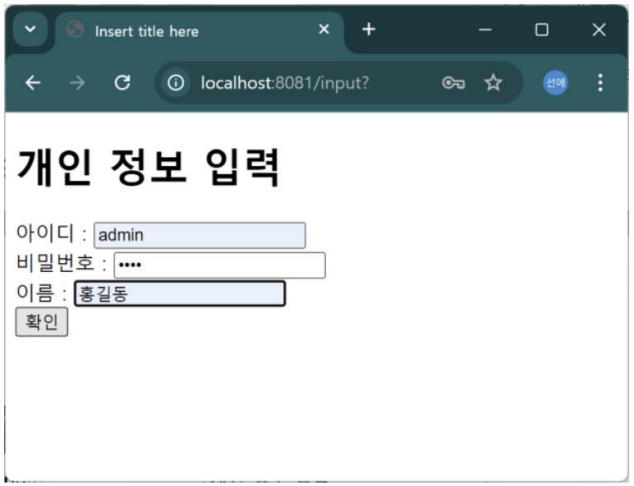
```jsp
<%@ page language="java" contentType="text/html;
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

<h1>개인 정보 입력 결과</h1>
<h3>
아이디 : ${id }<br>
패스워드 : ${pw }<br>
이름 : ${name }<br>
</h3>

</body>
</html>
```

실습(week11_03FormAndCommand)

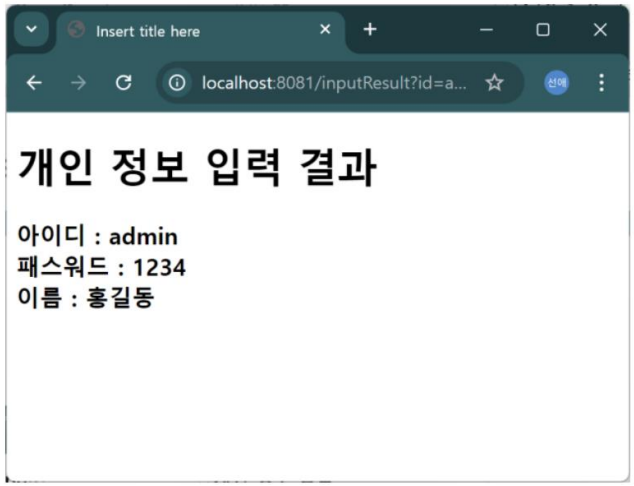
- **입력 폼** 사용
- **커맨드 객체**를 이용하여 **데이터를 뷰에 전달**



index.jsp



inputForm.jsp



inputResult.jsp

실습(week11_03FormAndCommand)

- **입력 폼** 사용
- **커맨드 객체**를 이용하여 **데이터를 뷰에 전달**
- **MemberInfo**

```java
package com.example.demo;

public class MemberInfo {
    private String id;
    private String password;
    private String name;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

실습(week11_03FormAndCommand)

- **입력 폼** 사용
- @RequestParam 어노테이션 사용
- **커맨드 객체**를 이용하여 **데이터를 뷰에 전달**
  - **MemberInfo 객체**

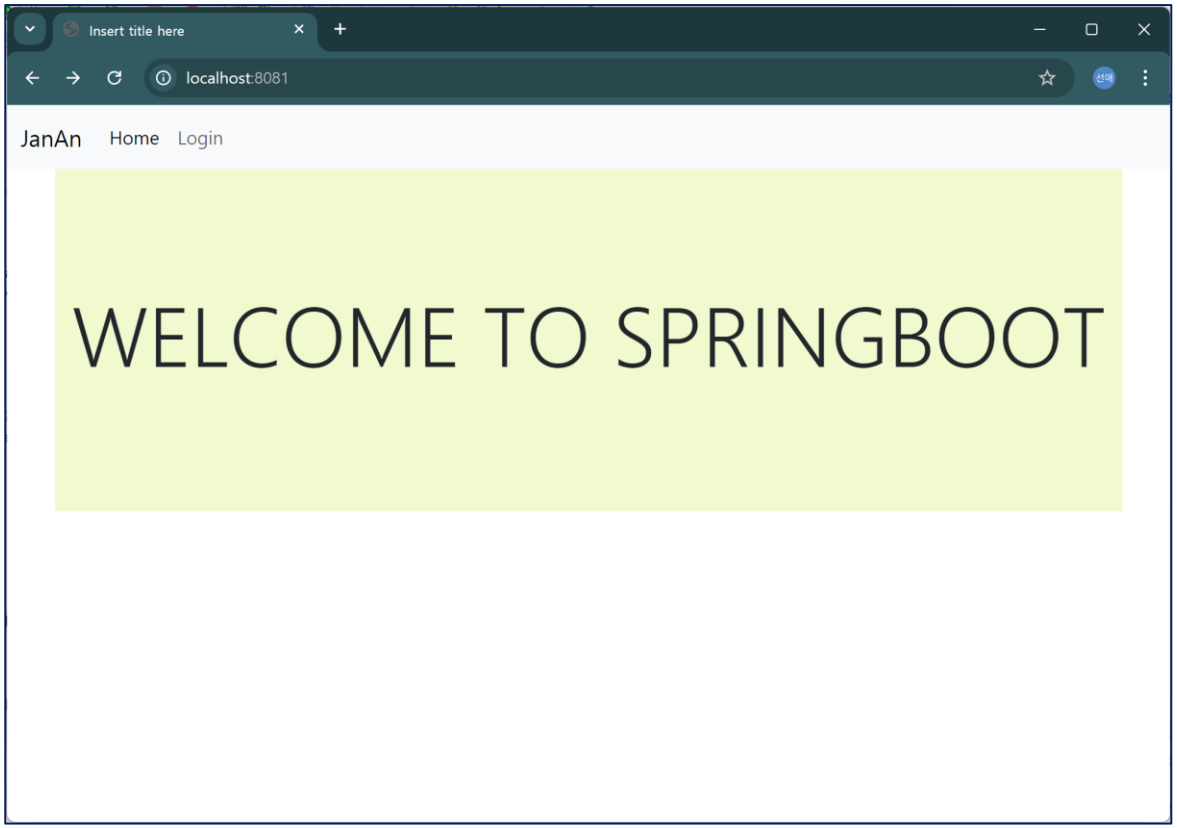
실습(week11_03FormAndCommand)

- **MyController**

```
@Controller
public class MyController {
    @RequestMapping("/")
    public String index() {
        return "index";
    }

    @GetMapping("/login")
    public String login() {
        return "loginForm";
    }

    @GetMapping("/loginResult")
    public String loginResult(MemberInfo memberInfo) {
        return "loginResult";
    }
}
```

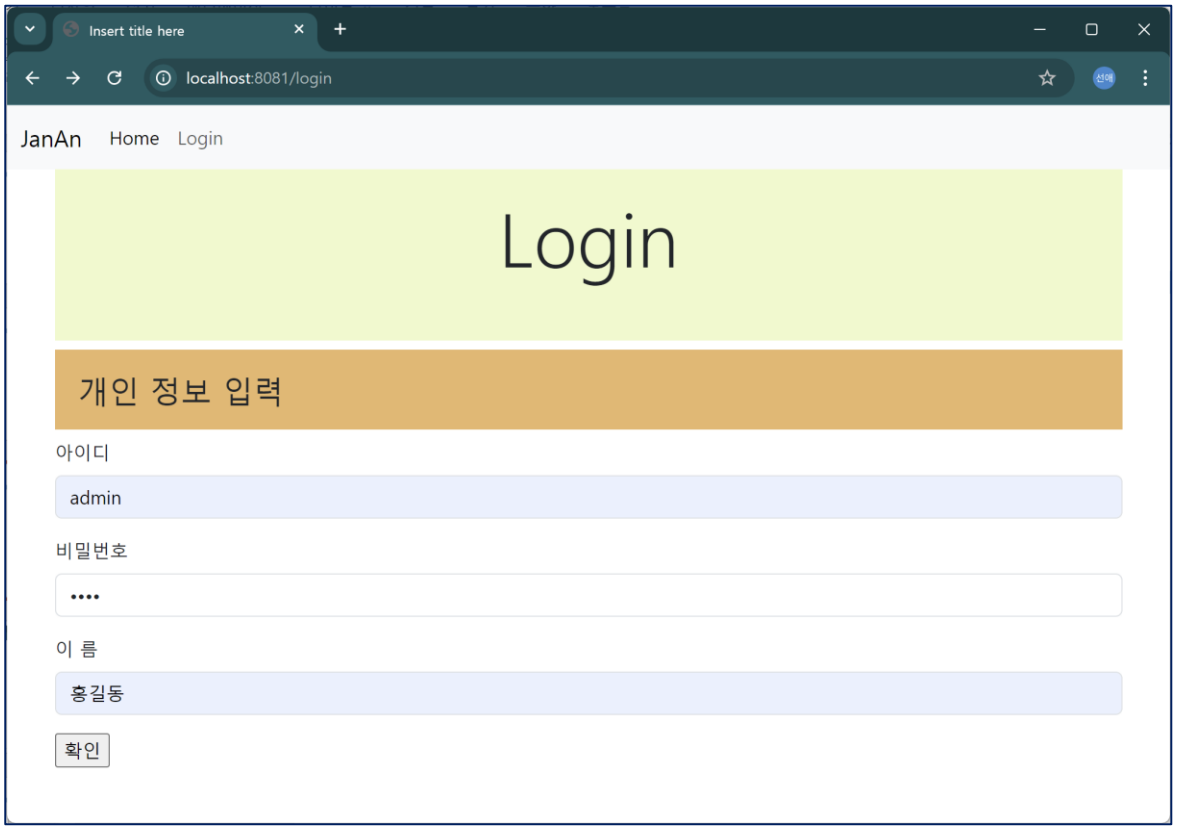
실습(week11_03FormAndCommand)

- 입력 폼 사용
- @RequestParam 어노테이션 사용
- Model 객체에 데이터 저장 후 뷰에 전달



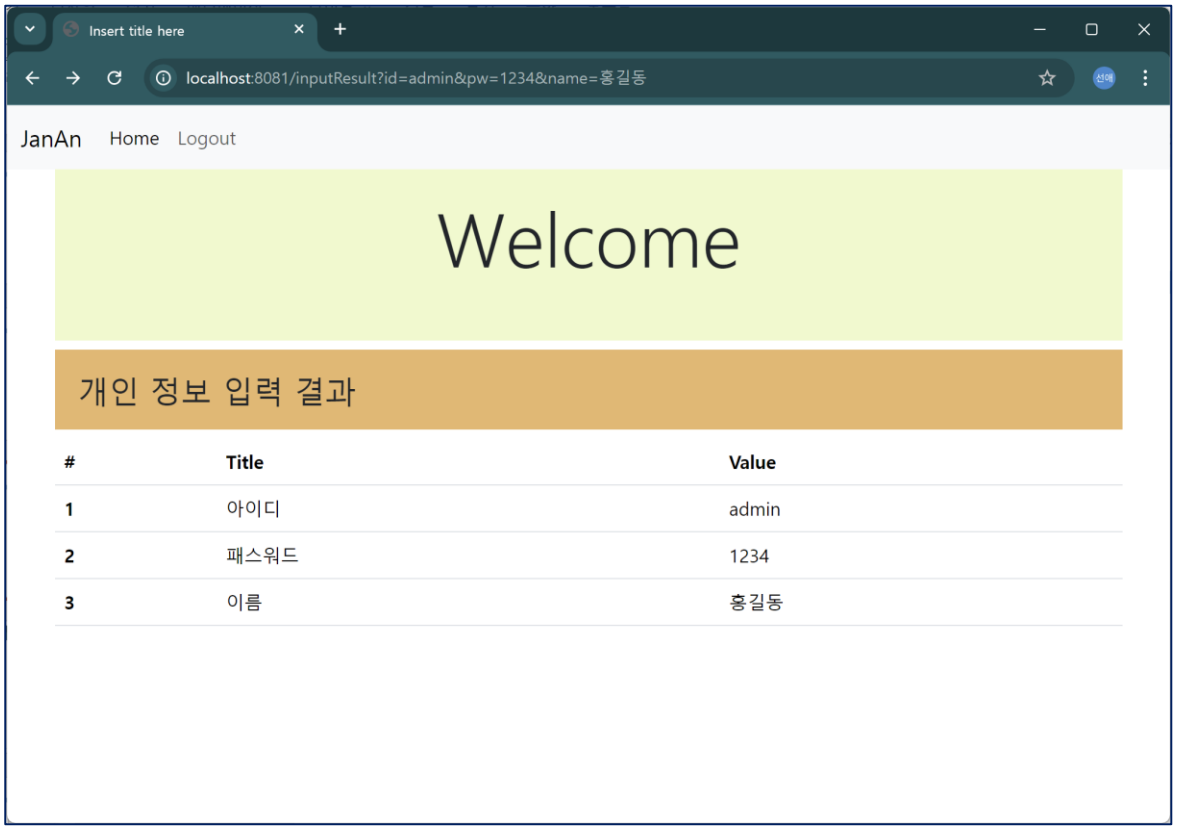
실습(week11_03FormAndCommand)

- 입력 폼 사용
- @RequestParam 어노테이션 사용
- Model 객체에 데이터 저장 후 뷰에 전달



실습(week11_03FormAndCommand)

- 입력 폼 사용
- @RequestParam 어노테이션 사용
- Model 객체에 데이터 저장 후 뷰에 전달



index.jsp

bootstrap 이용

index.jsp

```
7  <title>Insert title here</title>
8  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
9  integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
10 </head>
11 <body>
12 <nav class="navbar navbar-expand-lg bg-body-tertiary" >
13   <div class="container-fluid">
14     <a class="navbar-brand" href="/">JanAn</a>
15     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"
16         aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
17       <span class="navbar-toggler-icon"></span>
18     </button>
19     <div class="collapse navbar-collapse" id="navbarSupportedContent">
20       <ul class="navbar-nav me-auto mb-2 mb-lg-0">
21         <li class="nav-item">
22           <a class="nav-link active" aria-current="page" href="#">Home</a>
23         </li>
24         <li class="nav-item">
25           <a class="nav-link" href="/login">Login</a>
26         </li>
27       </ul>
28     </div>
29   </div>
30 </nav>
31 <div class="container text-center">
32     <h1 class="display-1" style="text-align: center;padding-top:100px; height: 300px; background-color: #effaca">
33         WELCOME TO SPRINGBOOT</h1>
34
35 </div>
36 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
37 integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>
38 </body>
```

bootstrap > 컴포넌트 > 내비게이션 바

loginForm.jsp(1)


```
<title>Insert title here</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
  integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
</head>
<body>
<nav class="navbar navbar-expand-lg bg-body-tertiary" >
  <div class="container-fluid">
    <a class="navbar-brand" href="/">JanAn</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/login">Login</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

bootstrap > 컴포넌트 > 내비게이션 바

loginForm.jsp(2)


```
</nav>
<div class="container text-center">
    <h2 class="display-2" style="text-align: center;padding-top:20px; height: 150px; background-color: #effaca">
        Login</h1>
    <p class="h3" style="text-align: left; height: 70px; padding-top:20px;padding-left:20px;
        background-color: #f0b669">개인 정보 입력</p>
</div>
<form action="/inputResult">
  <div class="container">
    <div class="mb-3">
      <label for="exampleFormControlInput1" class="form-label">아이디</label>
      <input type="text" class="form-control" name="id" id="exampleFormControlInput1">
    </div>
    <div class="mb-3">
      <label for="inputPassword5" class="form-label">비밀번호</label>
      <input type="password" id="inputPassword5" name="pw" class="form-control" aria-describedby="passwordHelpBlock">
    </div>
    <div class="mb-3">
      <label for="exampleFormControlInput1" class="form-label">이 름</label>
      <input type="text" class="form-control" name="name" id="exampleFormControlInput1">
    </div>
    <input type="submit" value="확인">
  </div>
</form>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>

</body>
```

bootstrap > 폼 > 폼 컨트롤

inputResult.jsp(1)


```
<title>Insert title here</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
    rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
    crossorigin="anonymous">
</head>
<body>
<nav class="navbar navbar-expand-lg bg-body-tertiary" >
  <div class="container-fluid">
    <a class="navbar-brand" href="/">JanAn</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
        data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
        aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/logout">Logout</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

로그아웃 요청

inputResult.jsp(2)


```
<div class="container text-center">
    <h2 class="display-2" style="text-align: center;padding-top:20px; height: 150px;
        background-color: #effaca">Welcome</h2>
    <p class="h3" style="text-align: left; padding-left:20px; padding-top:20px; height: 70px;
        background-color: #f0b669">개인 정보 입력 결과</p>
</div>
<div class="container">
    <table class="table">
      <thead>
        <tr>
          <th scope="col">#</th>
          <th scope="col">Title</th>
          <th scope="col">Value</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <th scope="row">1</th>
          <td>아이디</td>
          <td>${id }</td>
        </tr>
        <tr>
          <th scope="row">2</th>
          <td>패스워드</td>
          <td>${pw }</td>
        </tr>
        <tr>
          <th scope="row">3</th>
          <td>이름</td>
          <td>${name }</td>
        </tr>
      </tbody>
    </table>
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigi
```

bootstrap > 컨텐츠 > 테이블

${memberInfo.id}

${memberInfo.password}

${memberInfo.name}

inputResult.jsp(1)


```
<title>Insert title here</title>
<link href="https://cdn.jsdelivr.net/npm/bo
    rel="stylesheet" integrity="sha384-QWTK
    crossorigin="anonymous">
</head>
<body>
<nav class="navbar navbar-expand-lg bg-body
  <div class="container-fluid">
    <a class="navbar-brand" href="/">JanAn<
    <button class="navbar-toggler" type="bu
        data-bs-target="#navbarSupportedCon
        aria-expanded="false" aria-label="T
      <span class="navbar-toggler-icon"></s
    </button>
    <div class="collapse navbar-collapse" i
      <ul class="navbar-nav me-auto mb-2 mb
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/logout">Logout</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

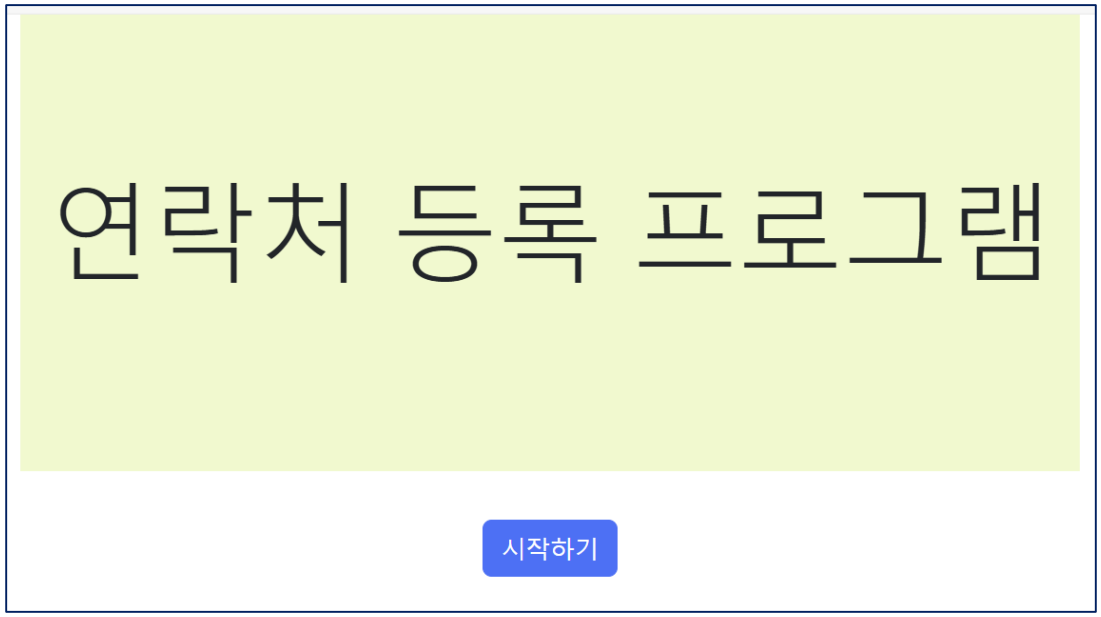
```
@GetMapping("/logout")
public String logout() {
    return "logoutForm";
}
```

```
<body>
    <!-- logoutForm.jsp -->
    <script type="text/javascript">
        if(confirm("정말 로그아웃하겠습니까?")){
            location.href = "/";
        }
    </script>
</body>
```

로그아웃 요청

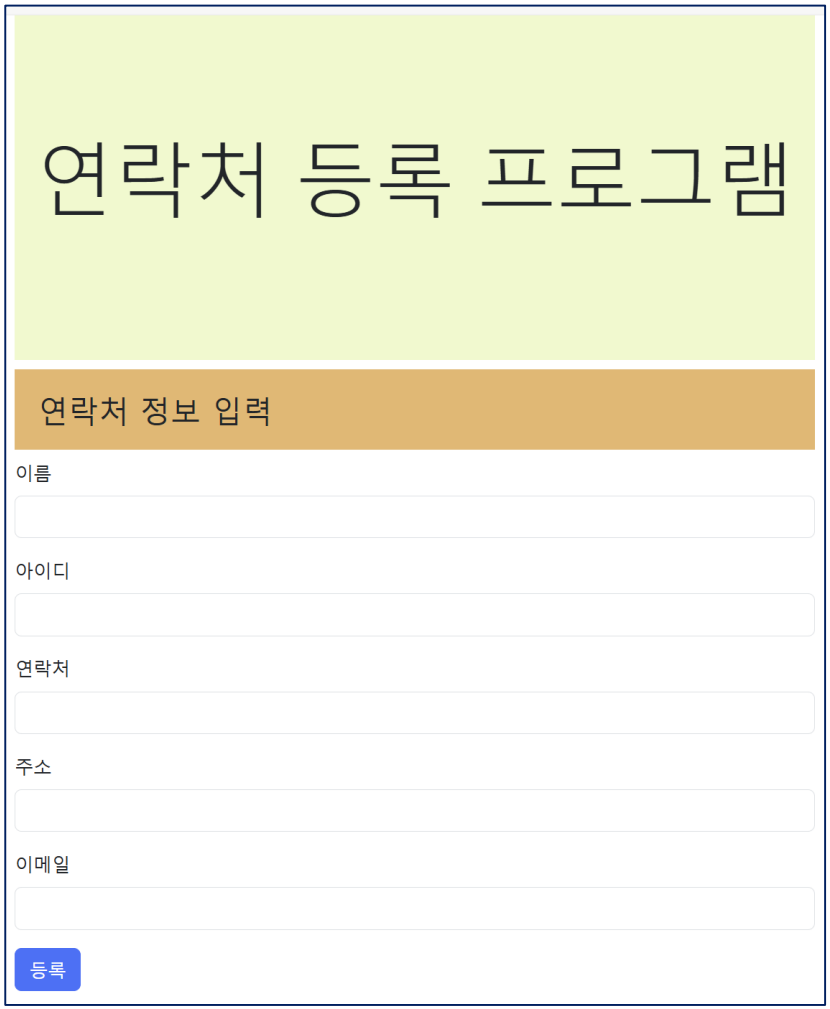
과제물 - 연락처 등록 프로그램

- 프로젝트명 : **week11_학번**
- 입력 폼 사용
- 커맨드 객체(Contact)를 사용해서 데이터를 뷰로 전달
  - 속성 : **name / id / phone / address / email**
- "/" 요청 시, "contactIndex.jsp"



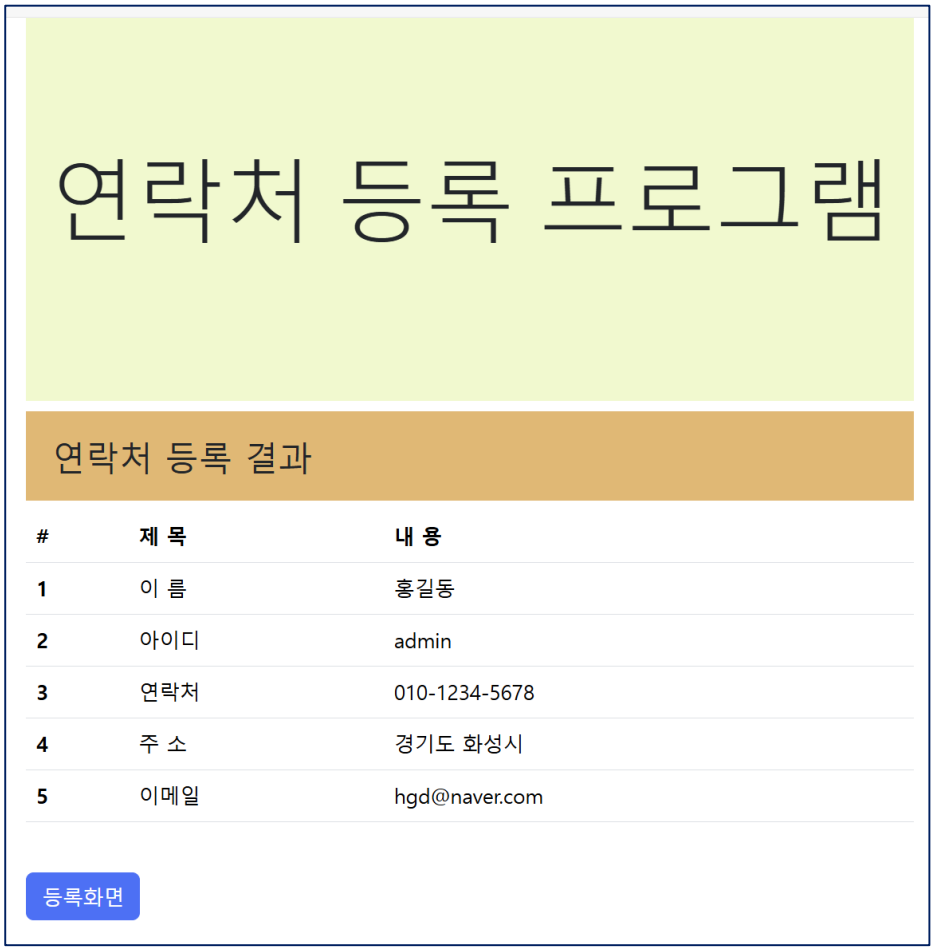
과제물 - 연락처 등록 프로그램

- "/contactForm" 요청 시, "contactForm.jsp"



과제물 - 연락처 등록 프로그램

- "/contactResult" 요청 시, "contactResult.jsp"



과제물 - 연락처 등록 프로그램

- 제출 형식
  - 프로젝트명 우클릭 > Export… > General > **Archive File**로 저장
  - **zip 파일** 제출
  - **최종 결과 화면 이미지** 파일 제출