

# 16장 모델2 방식으로 효율적으로 개발하기

---

1. 웹 애플리케이션 모델
2. MVC 디자인 패턴
3. 커맨드 패턴
4. MVC를 이용한 회원 관리

# 1. 웹 애플리케이션 모델

## 웹 애플리케이션 모델

- 애플리케이션 개발시 일반적으로 많이 사용하는 표준화된 소스 구조
- 모델의 종류에는 모델1과 모델2가 있음

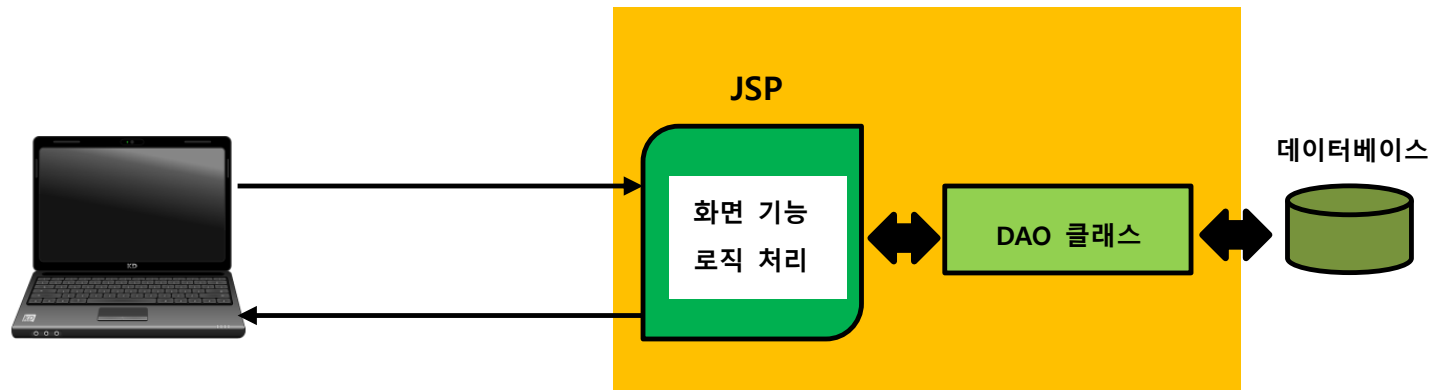
# 1. 웹 애플리케이션 모델

## 1.1 모델1 방식


### 모델1 방식


- 데이터베이스 연동 같은 비즈니스 로직 작업과 그 작업 결과를 나타내주는 작업을 동일한 JSP에서 수행함
- 모든 클라이언트의 요청과 비즈니스 로직 처리를 JSP가 담당하는 구조
- 기능 구현이 쉽고 편리하지만 유지보수가 어려움

### 모델1로 구현한 애플리케이션 동작 방식





# 1. 웹 애플리케이션 모델



**비스코탄탄가디건**  
예니추천 ♥  당일발송


판매가 16,900원  
적립금 170원 (1%)  
상품코드 P000FBGI


SNS 공유  


COLOR - [필수] COLOR 선택 -  
SIZE - [필수] SIZE 선택 -



(최소주문수량 1개 이상)


총 상품금액(수량) : 0 (0개)

 바로 구매하기  
BUY NOW

 장바구니 담기

 관심상품등록


**NAVER**  
네이버 ID로 간편구매  
네이버페이  Pay 구매 

신한포인트를 네이버페이로 전환하세요 

리뷰평점

4.9

베스트 포토리뷰



❖ 모델1으로 구현한 의류 쇼핑몰은 계절에 따라 화면도 업데이트를 해 줘야 하므로 유지 보수에 불편함

# 1. 웹 애플리케이션 모델

## 1.2 모델2 방식

### 모델2 방식

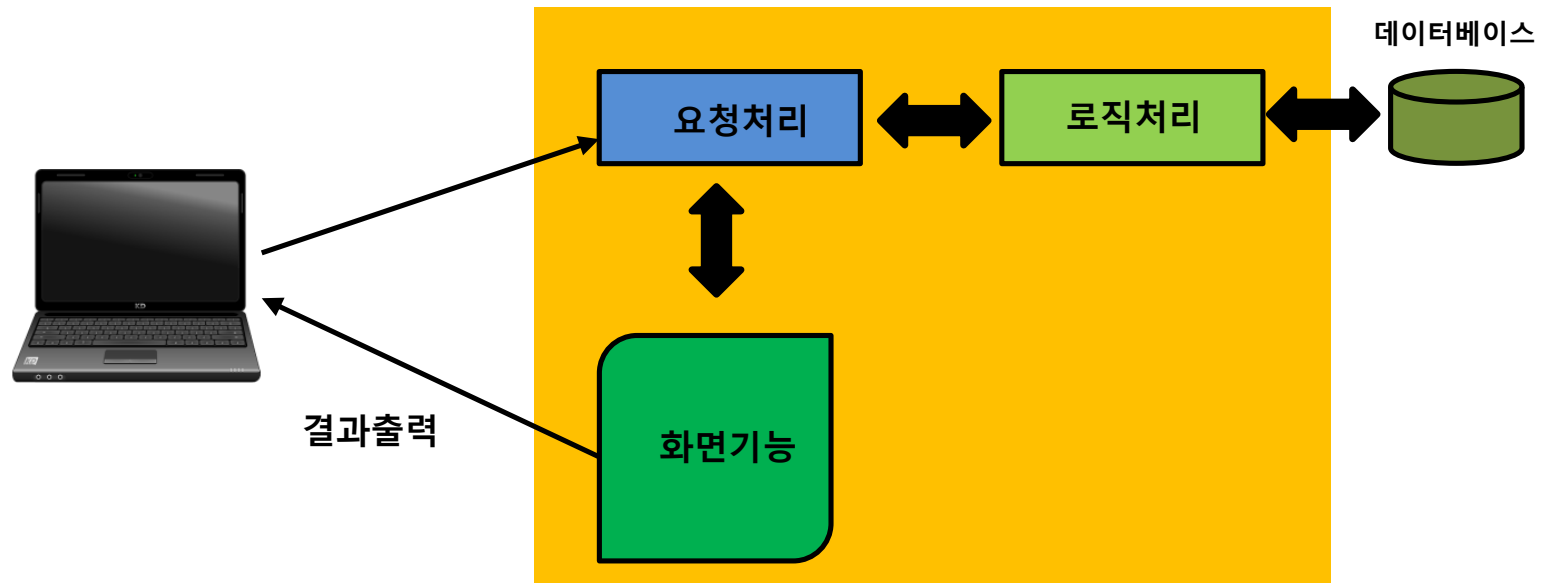
- 웹 애플리케이션의 각 기능(클라이언트의 요청 처리, 응답 처리, 비즈니스 로직 처리)을 분리해서 구현
- 객체 지향 프로그래밍에서 각각의 기능을 모듈화해서 개발하자는 원리

### 모델2 방식 특징

- 각 기능이 서로 분리되어 있어 개발 및 유지보수가 쉬움
- 각 기능(모듈)의 재사용성이 높음
- 디자이너와 개발자의 작업을 분업화해서 쉽게 개발할 수 있음
- 모델2 방식과 관련된 기능이나 개념의 학습이 필요

# 1. 웹 애플리케이션 모델

## 모델2 동작 방식



## 2. MVC 디자인 패턴

### 1.2 모델2 방식

#### MVC 패턴(Model-View-Controller pattern)

- 전통적인 GUI(Graphic User interface) 기반의 애플리케이션을 구현하기 위한 디자인 패턴
- MVC 구조는 사용자의 입력을 받아서 입력에 대한 처리를 하고, 그 결과를 다시 사용자에게 표시하기 위한 최적화된 설계를 제시

## 2. MVC 디자인 패턴

### Controller

- 서블릿이 컨트롤러의 역할
- 클라이언트의 요청을 분석
- 요청에 대해서 필요한 모델을 호출
- Model에서 처리한 결과를 보여주기 위해 JSP를 선택

### Model

- 데이터베이스 연동과 같은 비즈니스 로직을 수행
- 일반적으로 DAO와 VO 클래스로 이루어짐

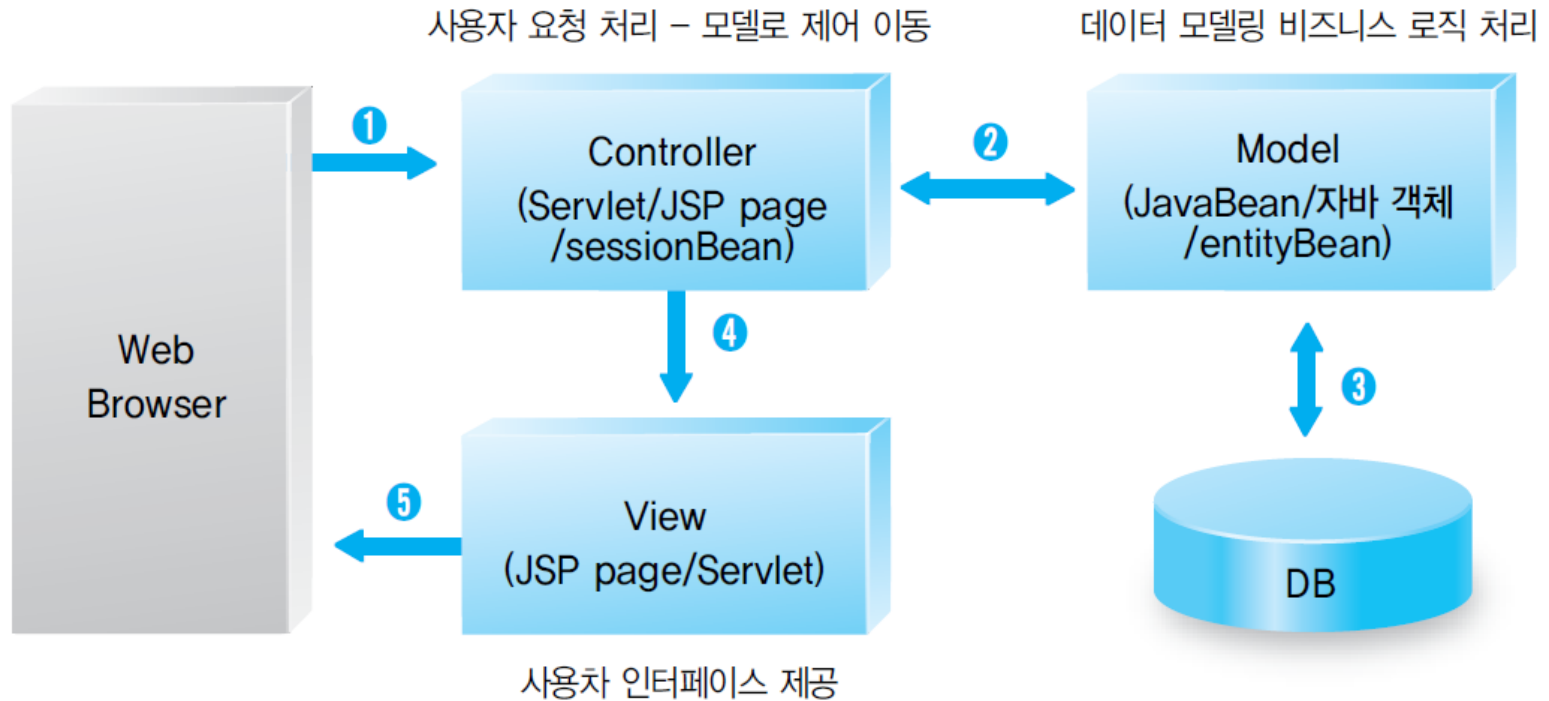
### View

- JSP가 화면 기능을 담당
- Model에서 처리한 결과를 화면에 표시



## 2. MVC 디자인 패턴

- MVC 패턴의 구조



## 2. MVC 디자인 패턴 -구성요소

### 컨트롤러(Controller) : 서블릿(Servlet)

- 웹브라우저의 요청을 받는 진입점
- 사용자의 요청을 받아서 요구사항을 분석 후 로직 처리를 모델로 보냄
- 로직의 처리 결과를 모델로부터 받아서, 사용자에게 응답하기 위해 뷰로 보냄

### 컨트롤러(Controller)의 작업 처리 과정

- ① 웹 브라우저의 요청을 받음
  - 웹브라우저의 요청은 서블릿의 서비스 메소드인 doGet() 또는 doPost()메소드가 받음
- ② 웹 브라우저가 요구하는 작업을 분석
  - 사용자가 요구한 작업에 맞는 로직이 실행되도록, 웹브라우저의 요구 작업을 분석
- ③ 모델을 사용해서 요청한 작업을 처리
  - 요청한 작업에 해당하는 로직을 처리
- ④ 로직 처리 결과를 request객체의 속성에 저장
  - request 객체의 속성에 처리 결과를 저장
  - 처리 결과는 같은 request 객체 영역에서 공유
- ⑤ 적당한 뷰(JSP페이지)를 선택 후 해당 뷰로 포워딩(forwarding).
  - 처리 결과를 저장한 request객체를 뷰로 전달

## 2. MVC 디자인 패턴 -구성요소

### 뷰(View) : JSP 페이지

- 요청에 대한 응답 결과를 표시
- JSP 페이지의 request는 컨트롤러(Controller)인 서블릿(Servlet)과 같은 객체로 공유
- `${requestScope.result}` 또는 `request.getAttribute("result")`와 같이 사용해서 결과를 화면에 표시

### 모델(Model) : 자바빈

- 컨트롤러(Controller)가 넘겨준 로직 처리
- 모델(Model)의 작업 처리과정
  - ① 컨트롤러(Controller)의 요청을 받음
  - ② 모델에서 로직을 처리
  - ③ 처리한 로직의 결과를 컨트롤러(Controller)로 반환

# 3. 커맨드 패턴

## 커맨드 패턴?

- 컨트롤러인 서블릿에 사용자의 요청을 명령어로 전달
- 사용자가 어떤 요청을 했는지 판단하기 위한 가장 일반적인 방법이 명령어로서 사용자의 요청을 전달
- 명령어와 로직을 연결하는 properties 매핑 파일이 필요

## 커맨드 패턴 종류

- 요청 파라미터로 명령어를 전달하는 방법
- 요청 URI 자체를 명령어로 사용하는 방법

### 3. 커맨드 패턴

#### 요청 파라미터로 명령어를 전달하는 방법

- 컨트롤러인 서블릿에 요청 파라미터를 정보를 덧붙여서 사용

예 : `http://localhost:8080/studyjsp/MessageContoller?message=aaa`

- 간편하긴 하나 명령어가 파라미터로 전달되게 되면 정보가 웹 브라우저를 통해 노출

#### 요청 URI 자체를 명령어로 사용하는 방법

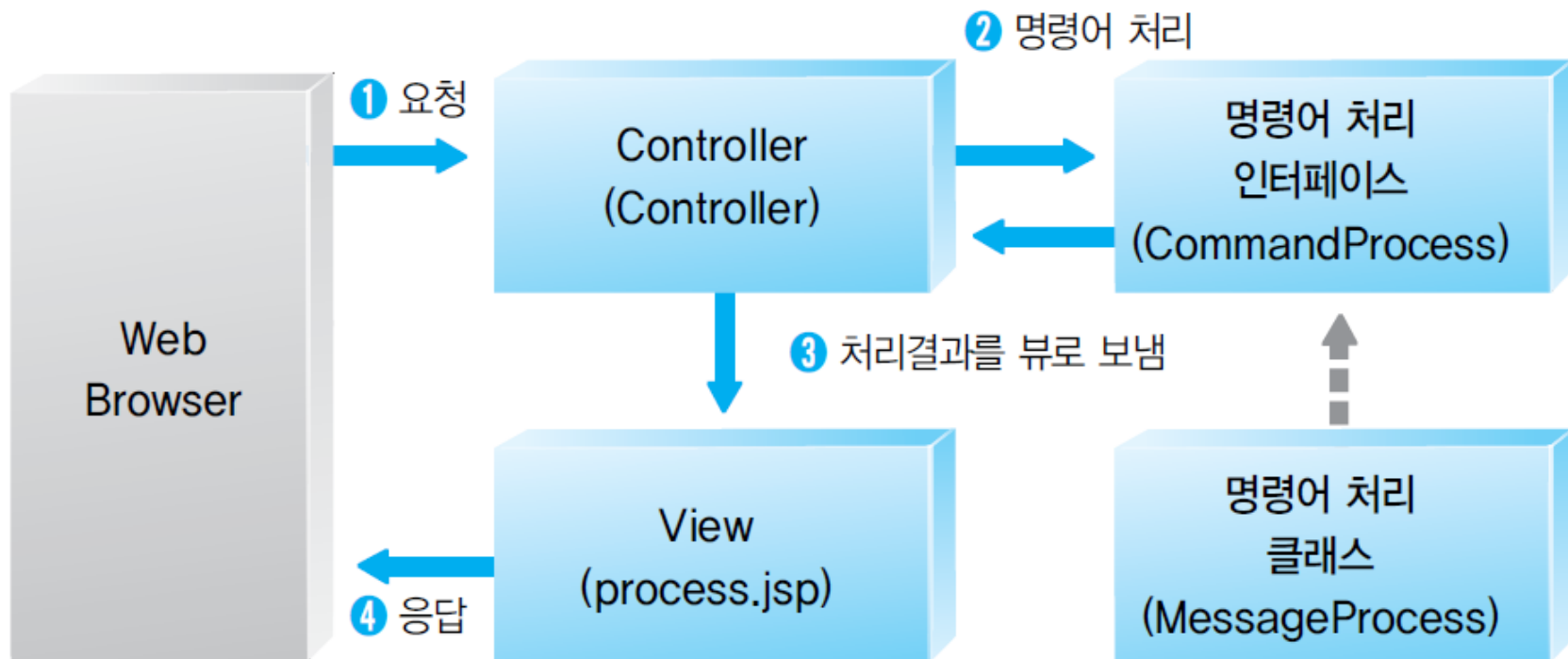
- 사용자가 요청한 URI 자체를 명령어로 사용하는 방법

예 : `http://127.0.0.1:8080/studyj네/ch17/test.do`

- 요청되는 URI가 실제 페이지가 아니고 명령어이므로 악의적인 명령어로부터 사이트가 보호되며, 요청되는 URL이 좀 더 자연스러워 진다는 장점

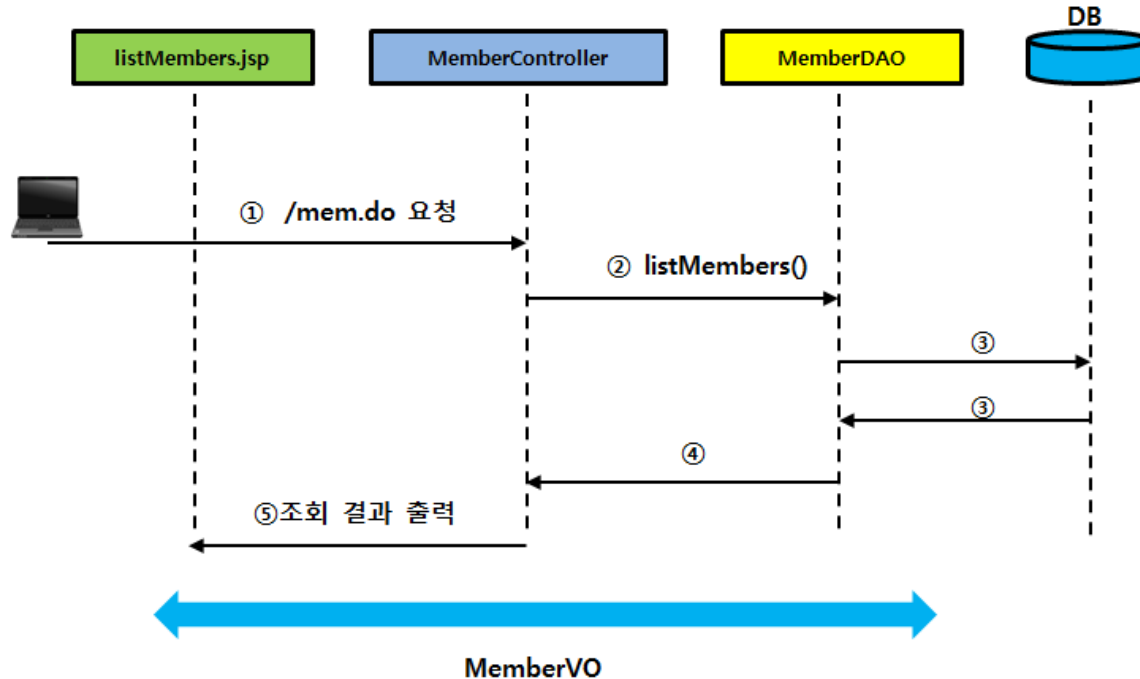
### 3. 커맨드 패턴

요청 파라미터로 전달하는 방법



# 3. MVC를 이용한 회원 관리

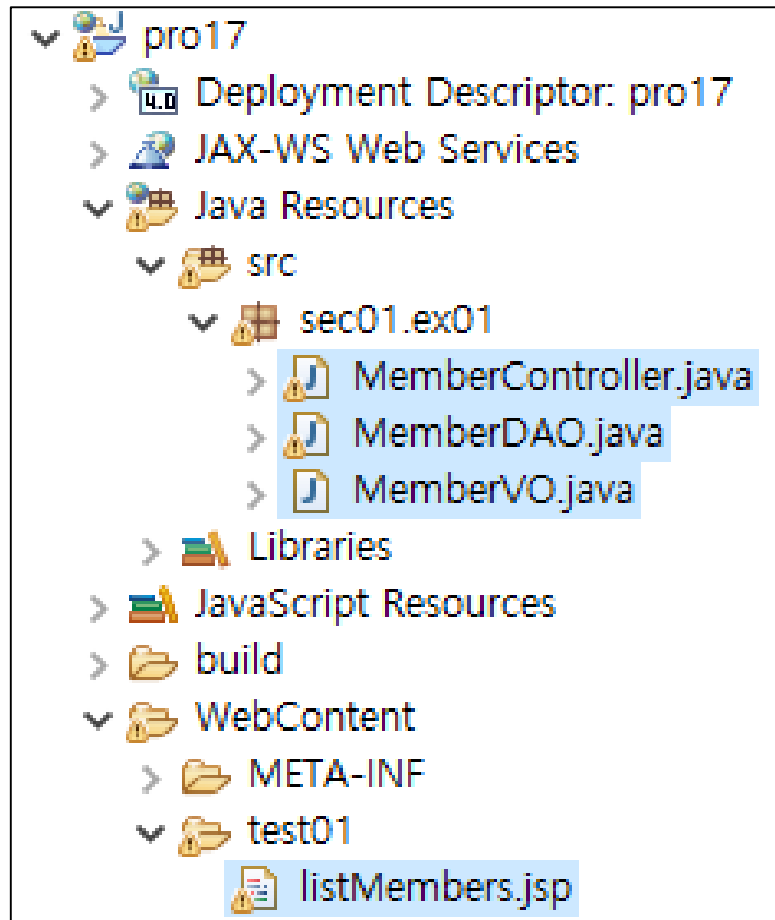
## 3.1 회원 정보 조회 기능 구현



- ① 브라우저에서 /mem.do로 요청
- ② 서블릿 MemberController가 요청을 받아 MemberDAO의 listMembers() 메서드를 호출
- ③ MemberDAO의 listMembers() 메서드에서 SQL문으로 회원 정보를 조회한 후 회원 정보를MemberVO에 설정하여 반환
- ④ 다시 MemberController에서는 조회한 회원 정보를 회원 목록창(listMembers.jsp)으로 포워딩
- ⑤ 회원 목록창(listMembers.jsp)에서 포워딩한 회원 정보를 목록으로 출력

### 3. MVC를 이용한 회원 관리

1. 새 프로젝트 pro17에 sec01.ex01 패키지를 만든 후 MemberController, MemberDAO, MemberVO 클래스를 추가합니다. 그리고 test01 폴더를 만들고 listMembers.jsp를 추가합니다.





### 3. MVC를 이용한 회원 관리

6. <http://localhost:8090/pro17/mem.do>로 요청하여 실행 결과를 확인합니다.

#### 회원정보

아이디	비밀번호	이름	이메일	가입일
ki	1234	기성용	ki@test.com	2018-09-13
kim	1212	김유신	kim@jweb.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

[회원 가입하기](#)

# 3. MVC를 이용한 회원 관리

## 3.2 회원 정보 추가 기능 구현

### 커맨드(Command) 패턴

- 브라우저가 URL 패턴을 이용해 컨트롤러에게 수행 작업을 요청하는 방법
- 컨트롤러는 HttpServletRequest의 getPathInfo() 메서드를 이용해 URL 패턴에서 요청명을 받아와 작업을 수행

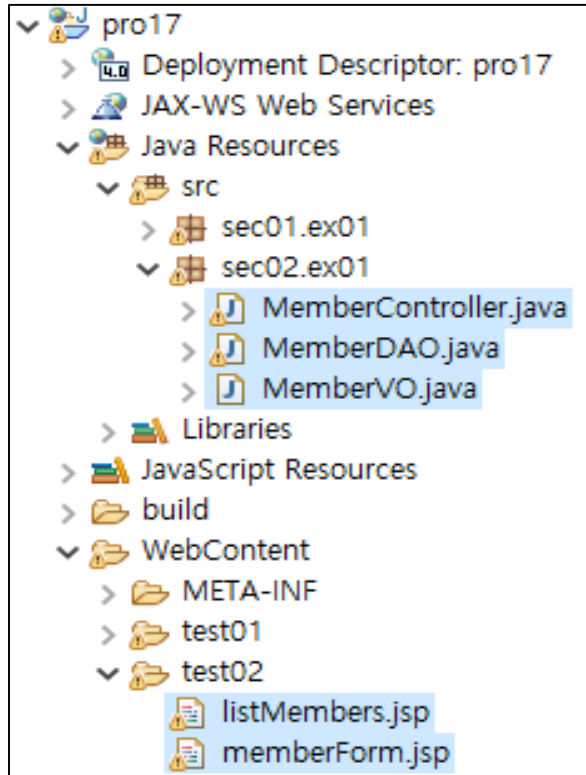
### URL 패턴을 이용해 컨트롤러에 요청하는 형식

- <http://localhost:8090/pro17/member/listMembers.do>

- ① /member : 회원 기능을 의미
- ② /listMembers.do : 회원 기능 중 회원 조회 기능을 의미

### 3. MVC를 이용한 회원 관리

1. sec02.ex01 패키지를 만들고 MemberDAO와 MemberVO 클래스는 sec01.ex01 패키지의 것을 복사해 붙여 넣습니다. 그리고 test01 폴더의 listMembers.jsp도 복사해 test02 폴더로 붙여 넣습니다.



# 3 MVC를 이용한 회원 관리

5. <http://localhost:8090/pro17/member/listMember.do>로 요청하여 회원 목록창이 나타나면 하단에 있는 회원 가입하기를 클릭합니다.

## 회원정보

아이디	비밀번호	이름	이메일	가입일
ki	1234	기성용	ki@test.com	2018-09-13
kim	1212	김유신	kim@jweb.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

[회원 가입하기](#)

6. 회원 가입창이 나타나면 다음과 같이 새 회원 차두리의 정보를 입력하고 가입하기를 클릭합니다.

7. 6번 과정에서 등록한 새 회원(차두리)이 추가된 회원 목록창이 다시 나타납니다.

## 회원 가입창

아이디

비밀번호

이름

이메일

## 회원정보

아이디	비밀번호	이름	이메일	가입일
cha2	1212	차두리	cha2@test.com	2018-11-22
ki	1234	기성용	ki@test.com	2018-09-13
kim	1212	김유신	kim@jweb.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

[회원 가입하기](#)

# 3 MVC를 이용한 회원 관리

## 3.3 회원 정보 수정 및 삭제 기능 구현

### 회원 정보 수정 과정

- ① 회원 정보 수정창에서 회원 정보를 수정하고 수정하기를 클릭해 /member/modMember.do로 컨트롤러에 요청
- ② 컨트롤러는 전송된 회원 수정 정보를 가져온 후 테이블에서 회원 정보를 수정
- ③ 수정을 마친 후 컨트롤러는 다시 회원 목록창을 표시

### 회원 정보 삭제 과정

- ① 회원 목록창에서 삭제를 클릭해 요청명 /member/delMember.do와 회원 ID를 컨트롤러로 전달
- ② 컨트롤러는 HttpServletRequest의 getPathInfo() 메서드를 이용해 요청명 얻음
- ③ 회원 ID를 SQL문으로 전달해 테이블에서 회원 정보를 삭제

### 3. MVC를 이용한 회원 관리

1. sec02.ex02 패키지를 만들고 앞에서 사용한 자바 실습 파일들을 붙여 넣습니다. 그리고 test03 폴더를 만들고 JSP 파일들을 붙여 넣습니다.

