

2020년도 1학기

자바실무프로젝트 과목

— 12주차 (보강완료되는 되는 주로서 13주차 완료) -

[교재 Day03-Class01-2 Model 아키텍처로 게시판 개발] p241~



- 1주차 ~ 4주차 : //스프링을 위한 선수 지식 : Servlet 요점 정리
- SpringProject1: (5주차) //개발자에 의한 객체 결합 (교재 p39~48)
- SpringProject2: (6주차) //스프링컨테이너에 의한 객체 결합 (XML 방식) - IoC- (교재 p49~91)
- SpringProject3: (7주차) //스프링컨테이너에 의한 객체 결합 (어노테이션 방식) - IoC- (교재 p93~108)
- SpringProject4_BizStep1: (8주차) //비즈니스 컴포넌트 실습 1 (게시판) -Business Layer- (교재 p109~127)
- SpringProject4_BizStep2: (9주차) // 비즈니스 컴포넌트 실습 2 (User추가) -Business Layer- (교재 p129~139)
- SpringProject5_AOP_step1: (9주차) // 스프링 AOP 개념 파악 - AOP - (교재 p143~170) 9주 과제
- SpringProject5_AOP_step2: (10주차) // AOP 동작 시점 (5개) 실습- AOP - (교재 p171~182) 10주 과제
- SpringProject5_AOP_step3: (10주차) // AOP 동작 시점과 JointPoint/바인드변수 실습- AOP - (교재 p183~192)
- SpringProject5_AOP_step4: (10주차) // 어노테이션 방식의 AOP 실습- AOP - (교재 p193~192)
- SpringProject6_Jdbc_step1: (11주차) // Spring JDBC -스프링 JDBC - (교재 p209~226) 11주 과제
- SpringProject6_Jdbc_step2: (11주차) // 트랜잭션 처리-트랜잭션 - (교재 p227~237)



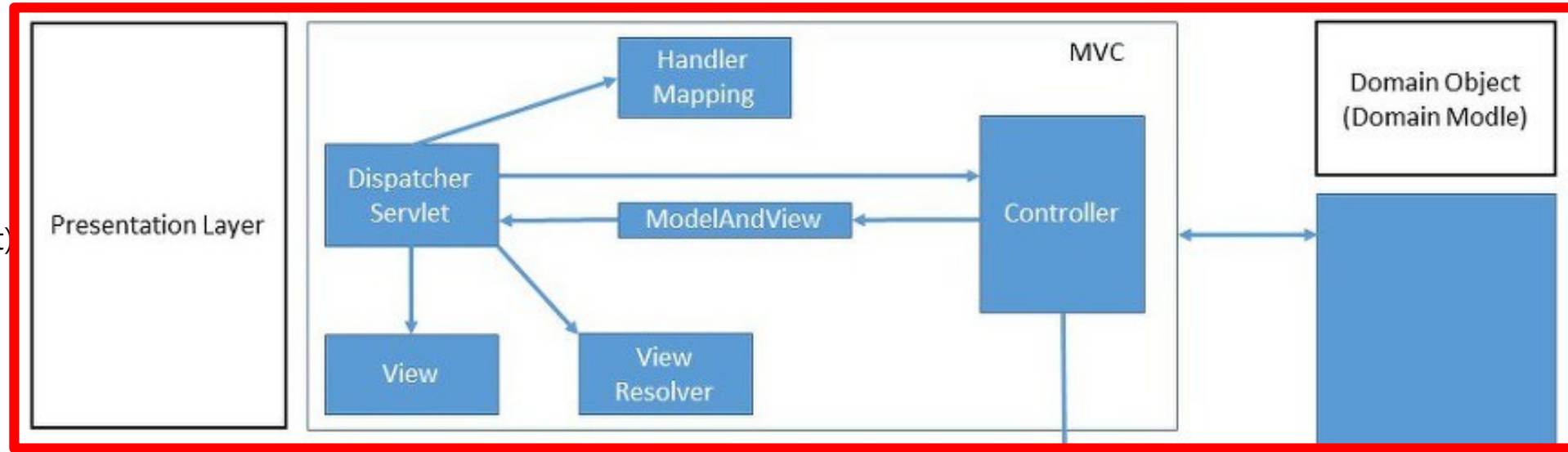
- SpringProject7_MVC_step1 : (12주차) // MVC 패턴 - Model 1 아키텍처 - (교재 p241~261)
- SpringProject7_MVC_step2 : (12주차) // MVC 패턴 - Model 2 아키텍처 - (교재 p263~283)
- SpringProject7_MVC_step3 : (12주차) // MVC 패턴 - MVC 프레임워크 아키텍처 - (교재 p285~311)
- SpringProject7_MVC_step4 : (12주차) // 스프링 MVC - Spring MVC 구조 (xml) - (교재 p313~345)



Spring Layered architecture 구조 – MVC 패턴 p241

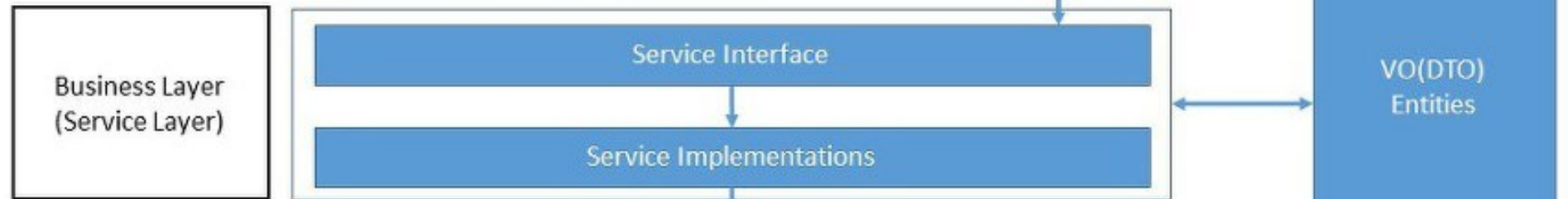
1) Presentation Layer

Spring MVC 객체를 말한다.
프론트 컨트롤러(DispatcherServlet),
컨트롤러, 뷰, 모델이 포함



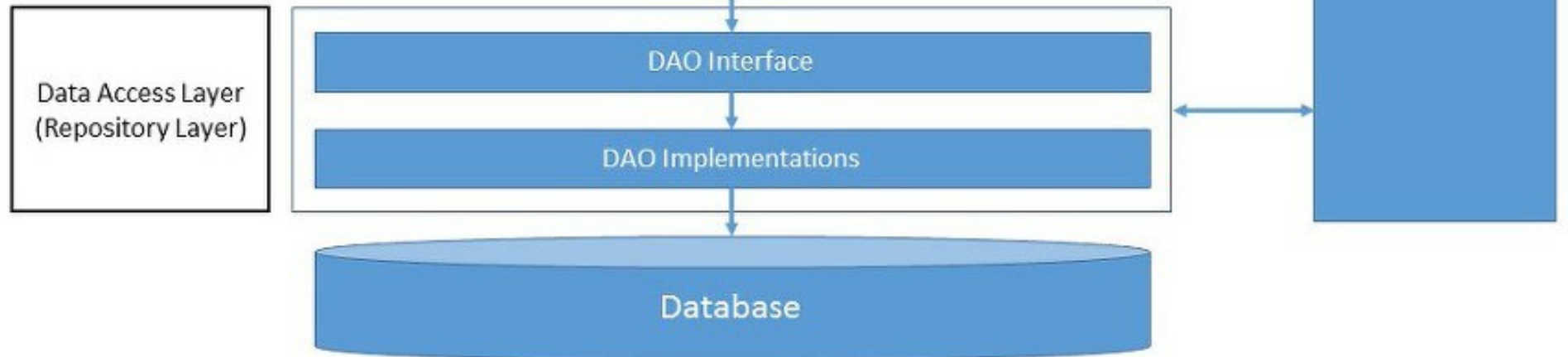
2) Service Layer(Business Layer)

presentation layer에서 요청을 보내면 실제로 비즈니스로직 수행



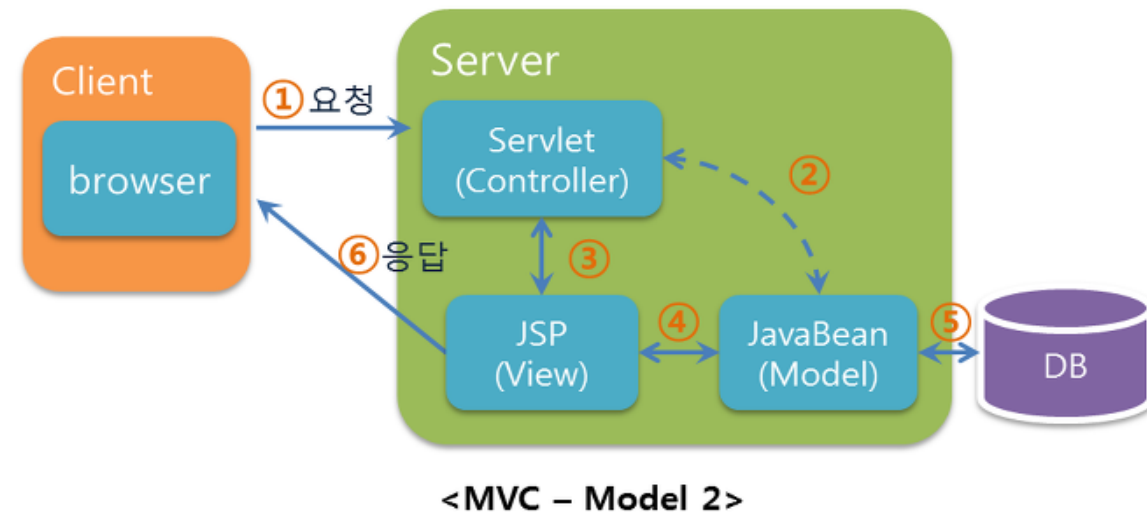
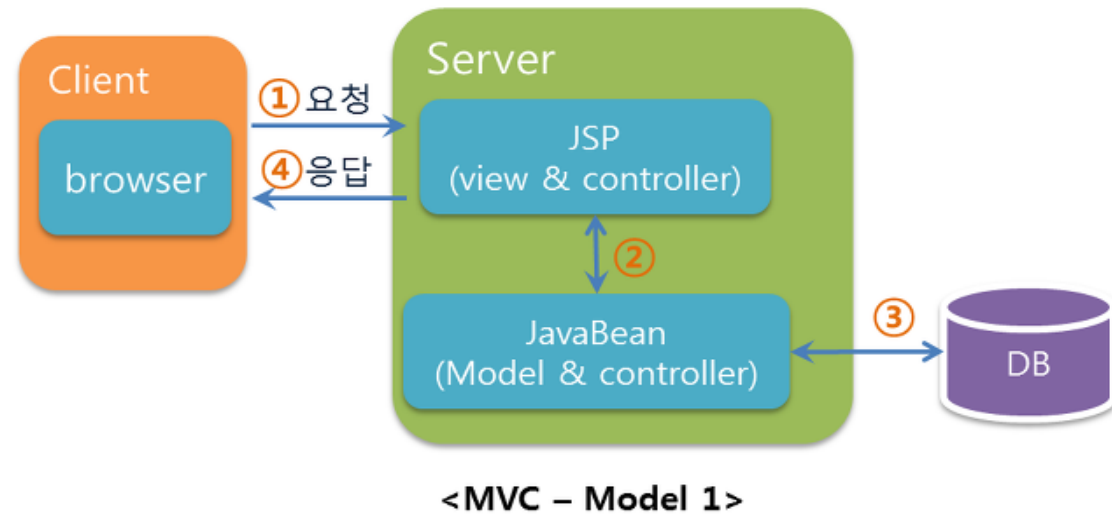
3) Data Access Layer(Repository Layer)

DB에 값을 저장하거나 가져오기 위해 JDBC, Mybatis, JPA 등을 사용해 구현한 DAO



• MVC 패턴 :

MVC패턴의 최대 장점은 사용자에게 보여지는 프레젠테이션 영역과 비즈니스 로직, 데이터 구조가 서로 완전히 분리되어 있다는 점



모델1: 비즈니스 로직 영역(Controller)에 프레젠테이션 영역(View)을 같이 구현하는 방식

모델2: 비즈니스 로직 영역과 프레젠테이션 영역이 분리되어 있는 구현 방식

	모델1	모델2
컨트롤러와 뷰의 분리 여부	통합(JSP파일)	분리(JSP, Servlet)
장점	쉽고 빠른 개발	디자이너/개발자 분업 유리 유지보수에 유리
단점	유지보수가 어려움	설계가 어려움 개발 난이도가 높음



- **모델 1** : 적은 개발인력으로 간단한 프로젝트 수행시 사용

- JSP

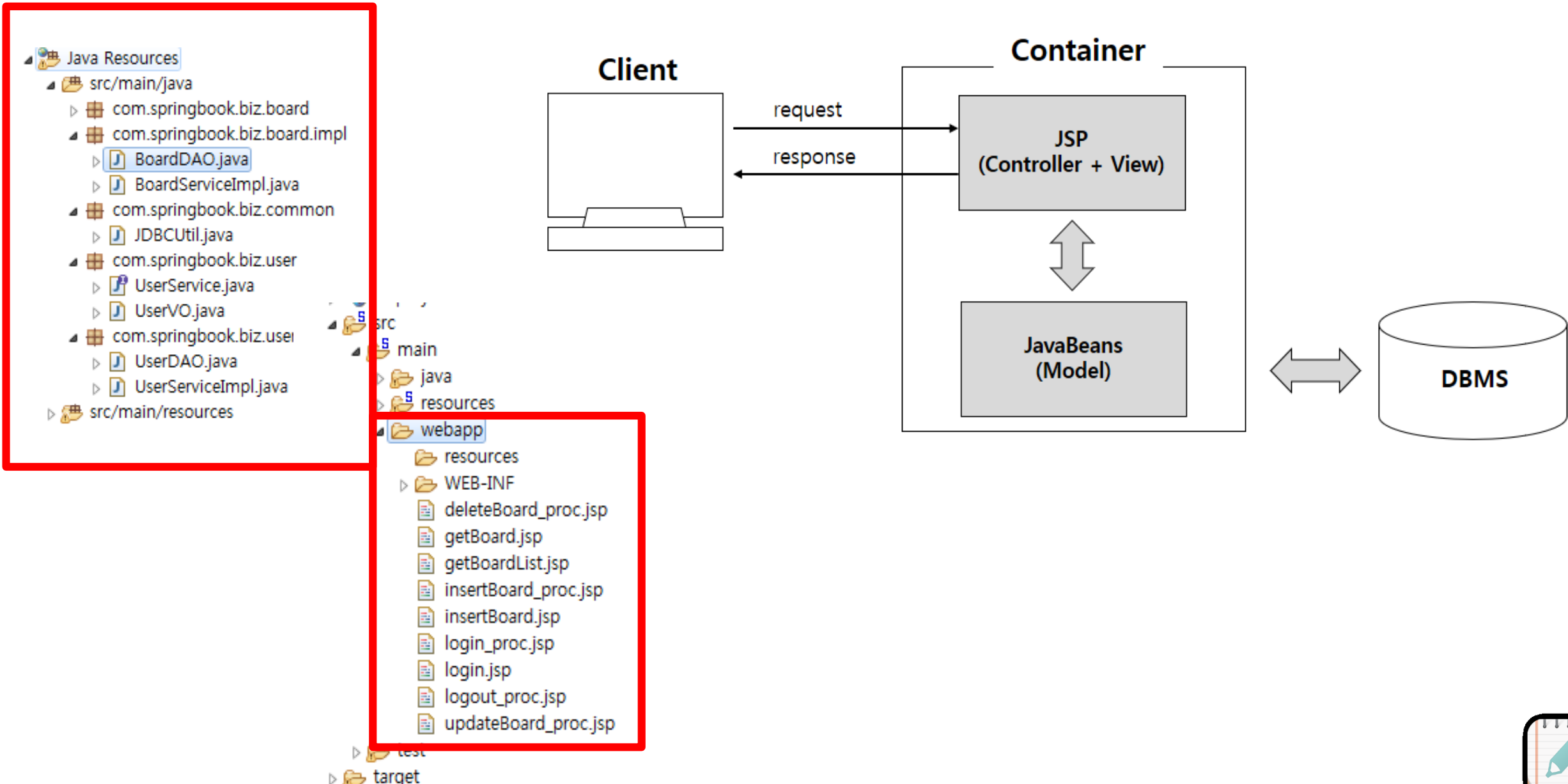
- **View 역할** : HTML, CSS
- **Controller 역할** : 사용자의 요청 처리와 관련된 자바 코드 의미 (jsp내의 모든 자바코드가 controller는 아님)

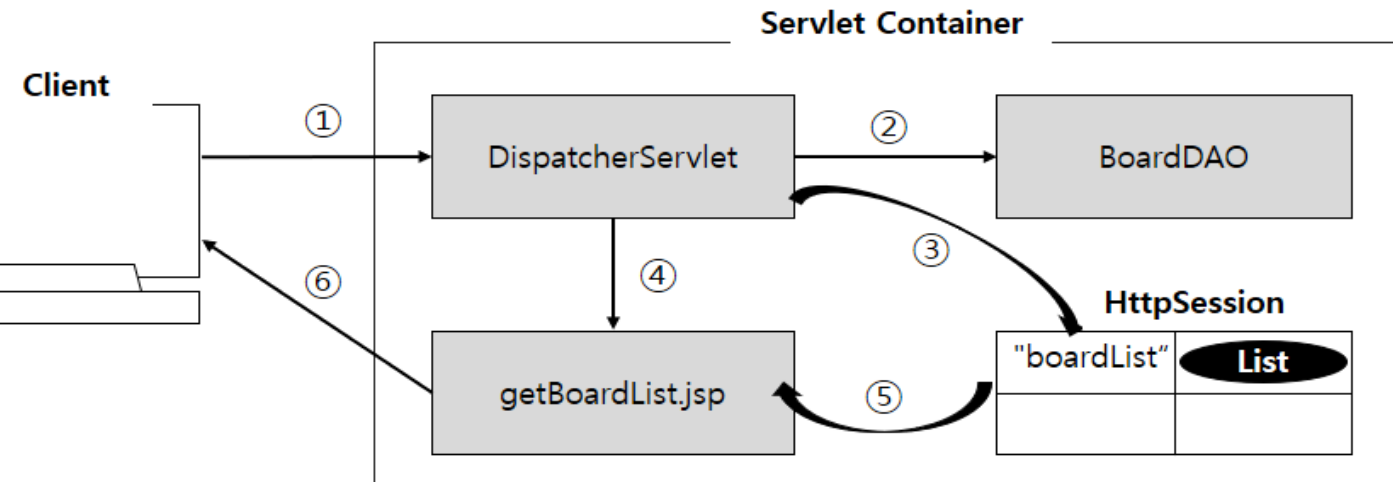
기 능	사 용 예
사용자 입력 정보 추출	<pre>String id = request.getParameter("userId")</pre>
DB 연동 처리	<pre>UserVO vo = new UserVO(); vo.setId(id); UserDAO userDAO = new UserDAO(); UserVO user = userDAO.getUser(vo);</pre>
화면 내비게이션	<pre>if(user != null) { // 로그인 성공 response.sendRedirect("getBoardList.jsp"); } else { // 로그인 실패 response.sendRedirect("login.jsp"); }</pre>

- JavaBean

- **Model 역할** : DB연동 로직을 제공하며, DB에서 검색한 데이터가 저장되는 자바 객체. VO와 DTO







기능	구성 요소	개발 주체
Model	VO, DAO 클래스	자바 개발자
View	JSP 페이지	웹 디자이너
Controller	Servlet 클래스	자바 개발자 또는 MVC 프레임워크

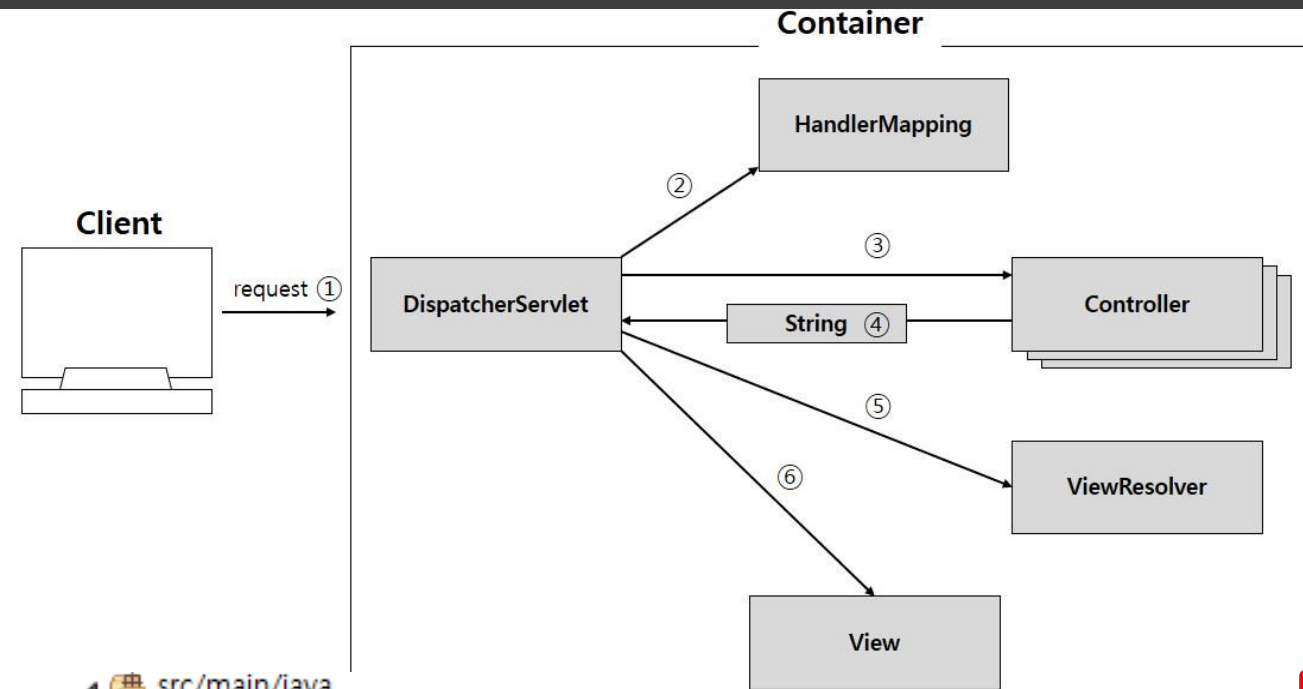
Java Resources

- src/main/java
 - com.springbook.biz.board
 - BoardDAO.java
 - BoardServiceImpl.java
 - com.springbook.biz.common
 - JDBCUtil.java
 - com.springbook.biz.user
 - UserService.java
 - UserVO.java
 - com.springbook.biz.user.impl
 - UserDAO.java
 - ServiceImpl.java
 - com.springbook.view.controller
 - DispatcherServlet.java
- src/main/resources

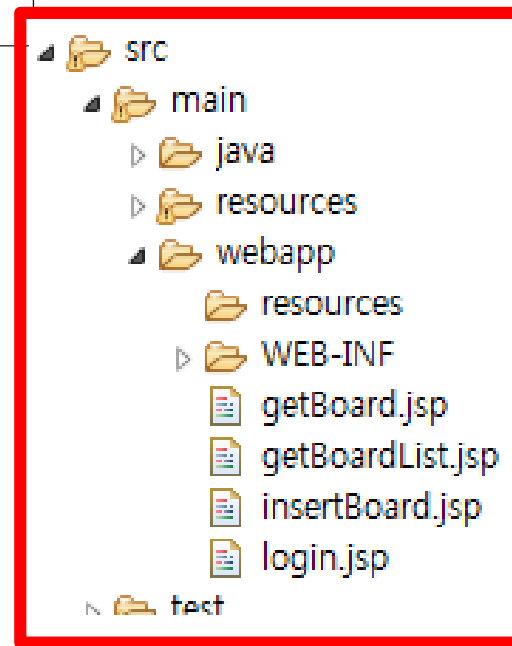
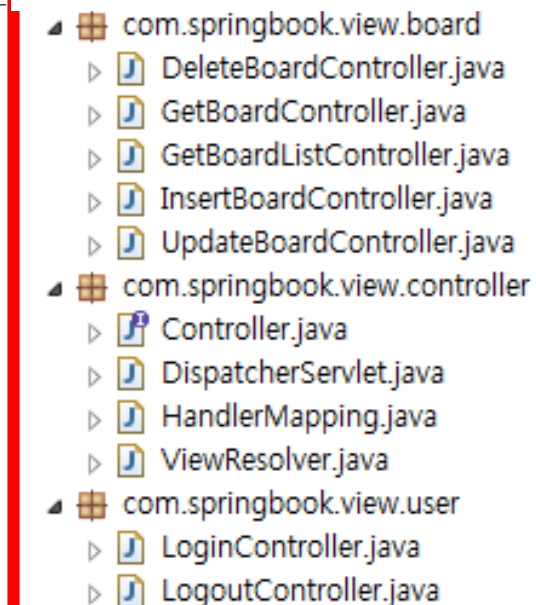
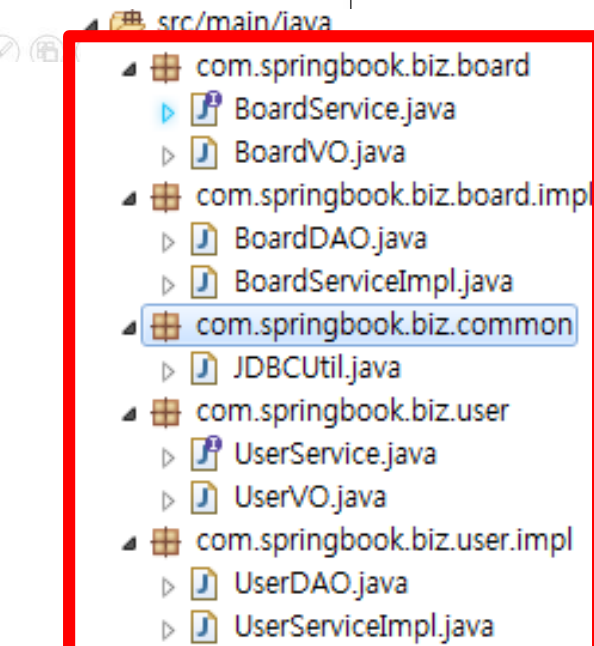
src

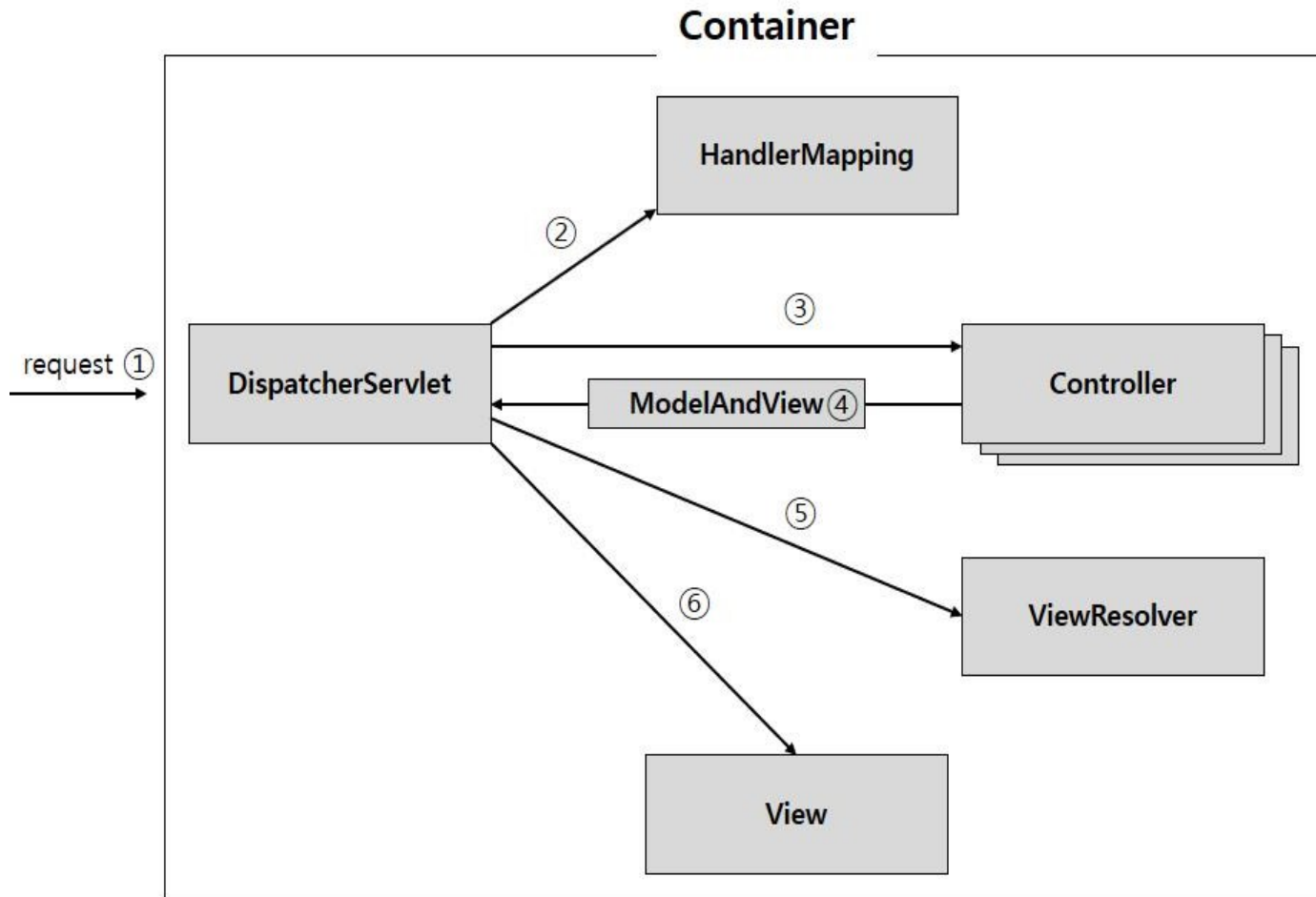
- main
 - java
 - resources
 - webapp
 - resources
 - WEB-INF
 - getBoard.jsp
 - getBoardList.jsp
 - insertBoard.jsp
 - login.jsp
- test





클래스	기능
DispatcherServlet	유일한 서블릿 클래스로서 모든 클라이언트의 요청을 가장 먼저 처리하는 Front Controller
HandlerMapping	클라이언트의 요청을 처리할 Controller 매핑
Controller	실질적인 클라이언트의 요청 처리
ViewResolver	Controller가 반환한 View 이름으로 실행될 JSP 경로 완성





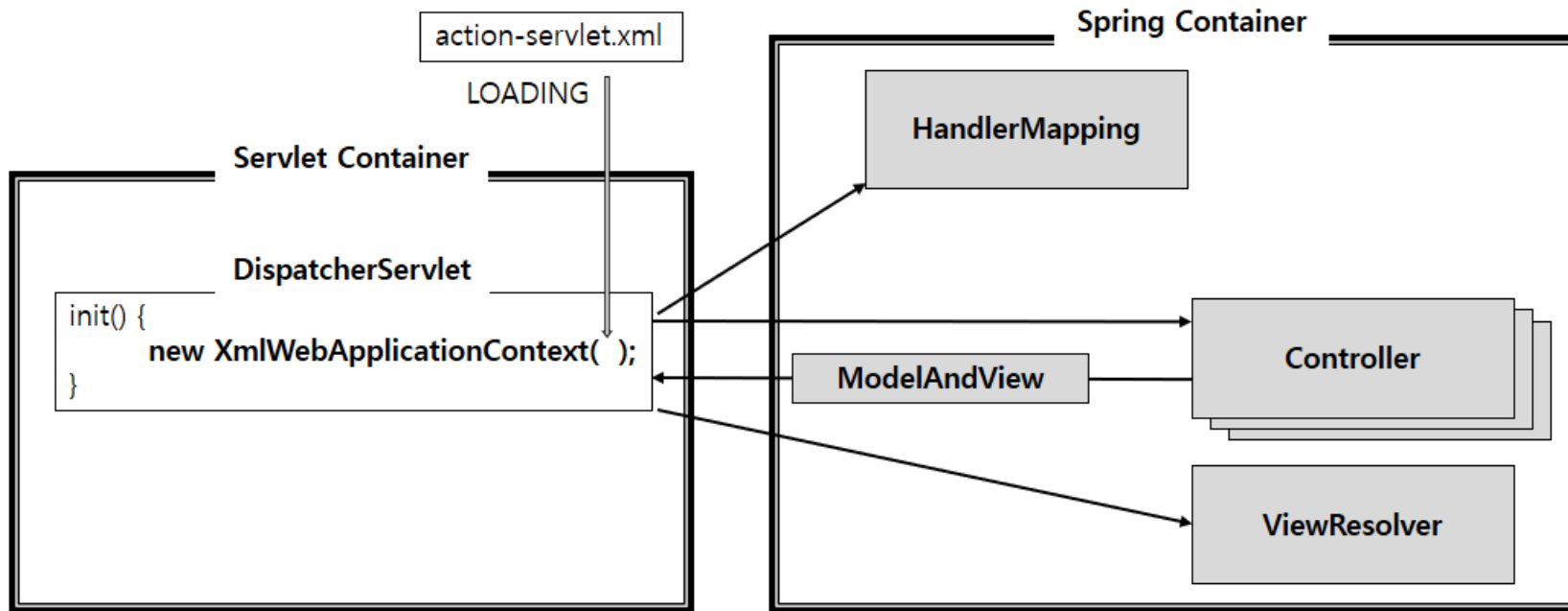
Java Resources

- src/main/java
 - com.springbook.biz.board
 - BoardService.java
 - BoardVO.java
 - com.springbook.biz.board.impl
 - BoardDAO.java
 - BoardServiceImpl.java
 - com.springbook.biz.common
 - JDBCUtil.java
 - com.springbook.biz.user
 - UserService.java
 - UserVO.java
 - com.springbook.biz.user.impl
 - UserDAO.java
 - UserServiceImpl.java
 - com.springbook.view.board
 - DeleteBoardController.java
 - GetBoardController.java
 - GetBoardListController.java
 - InsertBoardController.java
 - UpdateBoardController.java
 - com.springbook.view.user
 - LoginController.java
 - LogoutController.java

resources

- webapp
 - resources
 - WEB-INF
 - board
 - getBoard.jsp
 - getBoardList.jsp
 - classes
 - config
 - presentation-layer.xml
 - web.xml
 - insertBoard.jsp
 - login.jsp





DispatcherServlet 등록 (web.xml)

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>
    org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/config/presentation-layer.xml</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

Spring 설정파일 (presentation-layer.xml)

<!-- HandlerMapping 등록 -->

```
<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            <prop key="/login.do">login</prop>
        </props>
    </property>
</bean>
```

<!-- Controller 등록 -->

```
<bean id="login" class="com.springbook.view.user.LoginController"> </bean>
```

<!-- ViewResolver 등록 -->

```
<bean id="viewResolver"
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/board/"> </property>
    <property name="suffix" value=".jsp"> </property>
</bean>
```



Controller 클래스 구현

```
public class LoginController implements Controller {
```

```
    @Override
```

```
    public ModelAndView handleRequest(HttpServletRequest request,  
                                     HttpServletResponse response) {
```

```
        // 1. 사용자 정보 받아오기
```

```
        // 2. DB 연결
```

```
        // 3. 화면 정보 받아오기
```

```
        ModelAndView mav = new ModelAndView();
```

```
        if(user != null) {
```

```
            mav.setViewName("redirect:getBoardList.do");
```

```
        } else {
```

```
            mav.setViewName("login");
```

```
        }
```

```
        return mav;
```

```
    }
```

```
}
```

xxx.do로 이동할 때는 ViewResolver를 적용하기 않기 위해 "redirect:"을 View 이름 앞에 붙인다.

xxx.jsp로 이동할 때는 확장자 ".jsp"를 제거한다.



게시판 글 목록

```
public class GetBoardListController implements Controller {

    @Override
    public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response) {

        // 1. DB 연      결
        BoardVO vo = new BoardVO();
        BoardDAO boardDAO = new BoardDAO();
        List<BoardVO> boardList = boardDAO.getBoardList(vo);

        // 2. DB 쿼리 결과와 화면 정      결 ModelAndView에 저장하여 리턴
        ModelAndView mav = new ModelAndView();
        mav.addObject("boardList", boardList); // Model 정      결
        mav.setViewName("getBoardList.jsp"); // View 정      결
        return mav;

    }

}
```

