1. MVC패턴 (Model - View - Controller)

1. Model - 프로그램의 내부 상태, 데이터(정보) 를 뜻함
2. View - 사용자 인터페이스, 클라이언트에게 보여지는 화면을 뜻함
3. Controller - 데이터와 비즈니스 로직 간의 상호 작용을 뜻함, DB 접근 경우에 따라 Service에 접근

2. MVC1

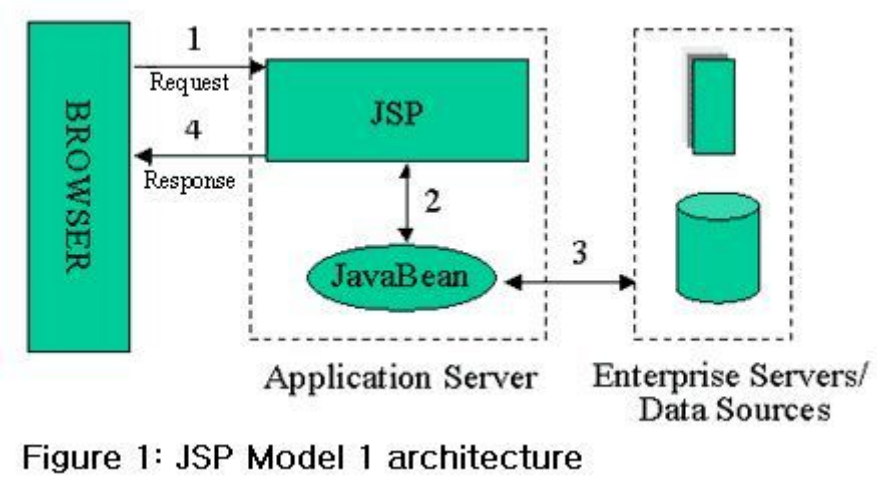


Figure 1: JSP Model 1 architecture

- JSP로 구현한 기존 웹 어플리케이션은 모델 1 구조로 웹 브라우저의 요청을 JSP 페이지가 받아서 처리 하는 구조이다.
- JSP 페이지에 비지니스 로직을 처리 하기 위한 코드와 웹 브라우저에 결과를 보여주기 위한 출력 관리 코드가 뒤섞여 있는 구조
- JSP 페이지 안에서 모든 정보를 표현(view)하고 저장(model)하고 처리(control)되므로 재사용이 힘들고, 읽기도 힘들어 가독성이 떨어진다.
- 정의: 모든 클라이언트 요청과 응답을 JSP가 담당하는 구조

3. MVC2

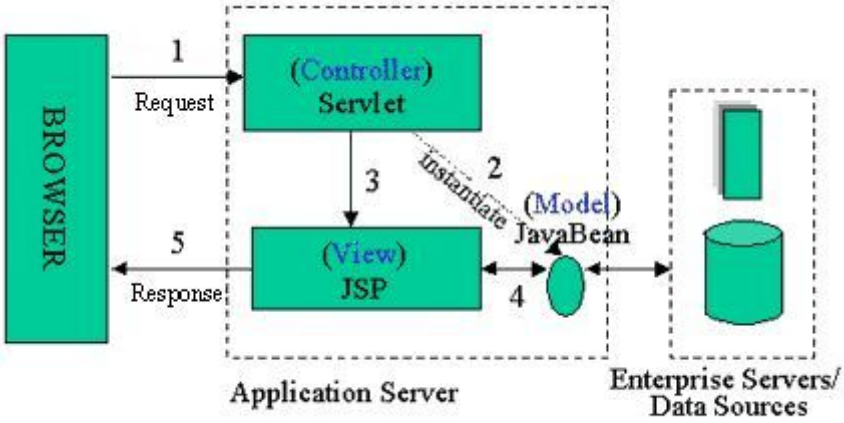
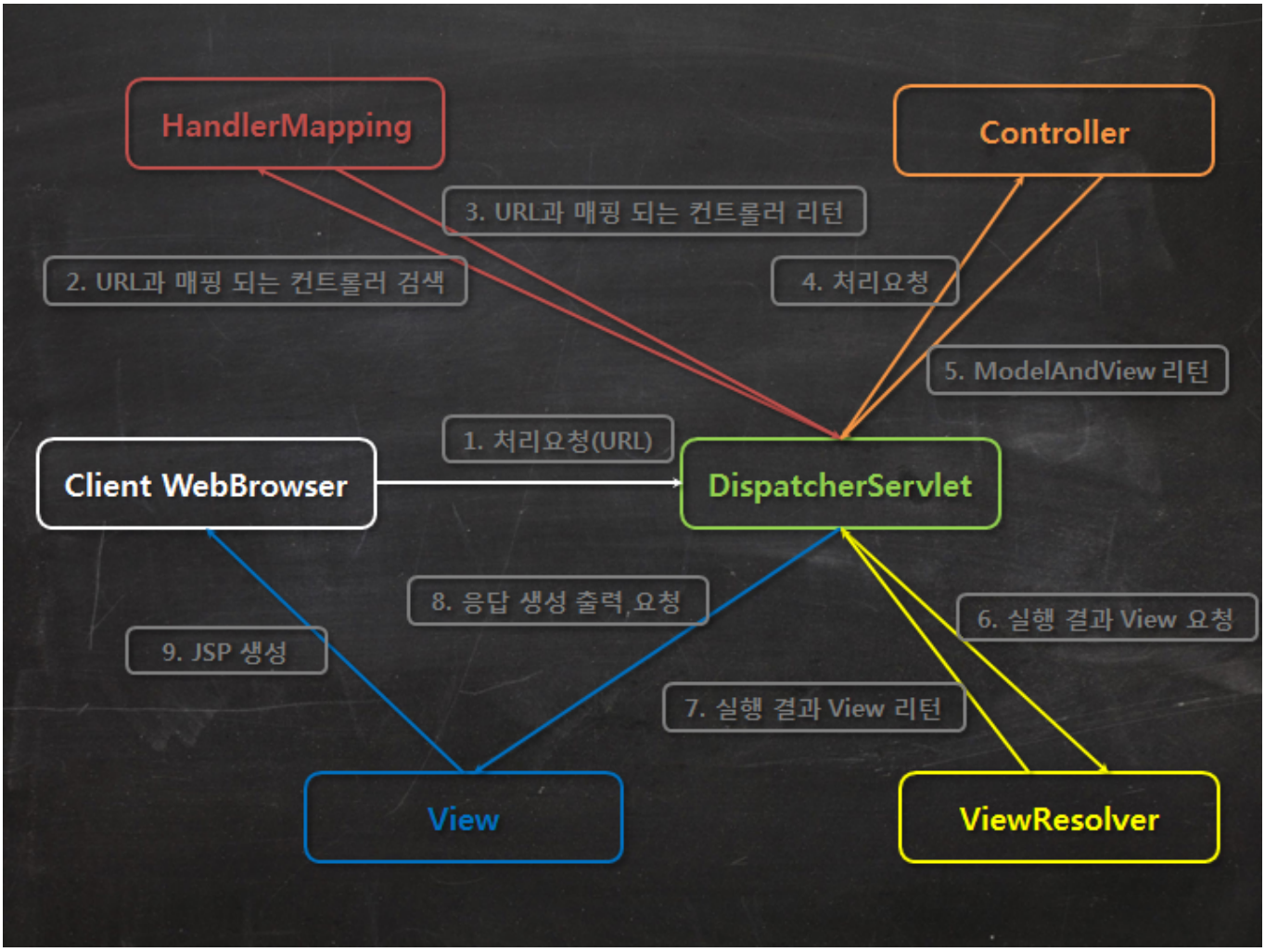


Figure 2: JSP Model 2 architecture

- MVC1 구조와 달리 웹 브라우저의 요청을 하나의 서블릿이 받게 됨
- 서블릿은 웹 브라우저의 요청을 알맞게 처리한 후 그 결과를 JSP 페이지로 포워딩
- 정의: 클라이언트의 요청처리와 응답처리, 비지니스 로직 처리하는 부분을 모듈화시킨 구조

4. Spring MVC



- MVC2 + 스프링 프레임워크
  - 스프링 프레임워크, 스프링이 제공하는 트랜잭션 처리, DI, AOP, 프레임워크를 손쉽게 사용
- MVC의 처리 순서

1. 클라이언트가 서버에 요청을 하면, front controller인 DispatcherServlet 클래스가 요청을 받는다
   - DispatcherServlet = web.xml
2. DispatcherServlet는 프로젝트 파일 내의 servlet-context.xml 파일의 @Controller 인자를 통해 등록한 요청 위임
   - < annotation-driven />
3. 컨트롤러를 찾아 매핑(mapping)된 컨트롤러가 존재하면 @RequestMapping을 통해 요청을 처리할 메소드로 이동한다.
4. 컨트롤러는 해당 요청을 처리할 Service(서비스)를 받아 비즈니스로직을 서비스에게 위임한다.
5. Service(서비스)는 요청에 필요한 작업을 수행하고, 요청에 대해 DB에 접근해야한다면 DAO에 요청하여 처리를 위임한다.
6. DAO는 DB정보를 DTO를 통해 받아 서비스에게 전달한다.
7. 서비스는 전달받은 데이터를 컨트롤러에게 전달한다.
8. 컨트롤러는 Model(모델) 객체에게 요청에 맞는 View(뷰) 정보를 담아 DispatcherServlet에게 전송한다.
9. DispatcherServlet는 ViewResolver에게 전달받은 View정보를 전달한다.
   - ViewResolver = InternalResourceViewResolver
10. ViewResolver는 응답할 View에 대한 JSP를 찾아 DispatcherServlet에게 전달한다.
11. DispatcherServlet는 응답할 뷰의 Render를 지시하고 뷰는 로직을 처리한다.
12. DispatcherServlet는 클라이언트에게 Rendering된 뷰를 응답하며 요청을 마친다