Framework

MVC(모델, 뷰, 컨트롤러) & 프레임워크

Model1과 Model2 방식

- 웹 애플리케이션을 작성하기 위해서는 실제 업무를 처리하는 부분(비즈니스 로직)과 화면을 처리하기 위한 부분(Presentation 로직) 부분을 구현해야 합니다.

항목	설명
비즈니스 로직	실제 데이터베이스와 연동하여 데이터를 얻어 오기 위한 작업을 수행하는 영역 Model(모델)
프리젠테이션 로직	최종 결과물을 클라이언트에게 보여 주는 역할을 수행하는 영역 View(뷰)

- 웹 애플리케이션 개발 방식의 종류
  - Model 1 방식(Model 1 Architecture)
  - Model 2 방식(Model 2 Architecture)_MVC 패턴

용어(Model)

- Database와 연동하는 비즈니스 로직을 통하여 얻어온 데이터를 저장하는 용도로 사용하는 데 보통 DAO(Data Access Object)라고 합니다.
- 실제 비즈니스 (업무) 로직이 구현되어 있습니다.
- View(보여 지는 영역)에게 제공할 데이터를 가공/관리합니다.
- 비즈니스 데이터와 데이터의 연산의 집합체의 결과물입니다.
- 일반적으로 DBMS(오라클, mysql)에 의해 관리됩니다.
  - DB와 관련된 연산은 JDBC API에 의존합니다.
  - 예시 : 게시판 글 등록/수정/삭제 등등(DBMS와 연동)입니다.
- EJB : 중대형 이상의 애플리케이션 모델의 한 종류를 말합니다.

- 예시
  - Java Bean(데이터 1건)
  - VO(Value Object) : 값을 저장하고 있는 객체
  - DTO(Data Transfer Object) : 데이터를 전송하는 객체
  - DAO(Data Access Object) : 데이터를 처리하는 객체

용어(View)

- 단순히 결과를 화면에 출력해 주는 부분(프리젠테이션 로직)입니다.
- 사용자와의 인터페이스로써 클라이언트에게 최종적으로 보여 지는 결과물 페이지입니다.
- JSP 페이지로 구성되는 것이 가장 일반적인 패턴입니다.

- 구현시 참조 기술
  - EL & JSTL
  - 커스텀 태그 라이브러리를 통한 로직 구현이 가능합니다.
  - Velocity 또는 XML/XSLT등 활용이 가능합니다.
- 예시
  - JSP 파일, HTML 파일
  - Excel, Poperpoint, Ms-Word 문서
  - PDF 파일 Format

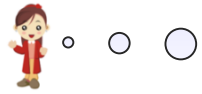
- 참조 사항
  - JSF(JavaServerFaces) : View 영역에 초점을 맞춘 프레임워크

용어(Controller)

- 사용자(웹 브라우저)의 요청 정보를 받아서 이것을 분석합니다.
- Model과 View 부분간의 흐름을 제어하는 역할입니다.
- 사용자의 요청을 처리할 자바 빈을 생성하고, 비즈니스 로직이 구현된 메소드를 실행합니다.
- 사용자의 요청에 따라 비즈니스 로직(View) 또는 Model 등을 호출하는 역할입니다.
- 비즈니스 로직을 수행 후 사용자의 요청을 JSP 페이지나 혹은 다른 URL으로 이동시킵니다.

- 구현시 참조 기술
  - 서블릿 맵핑(특정 페이지 요청에 대한 매핑 기능).

- 구현 예시
  - 사용자 정의 Servlet
  - Framework가 제공하는 Servlet

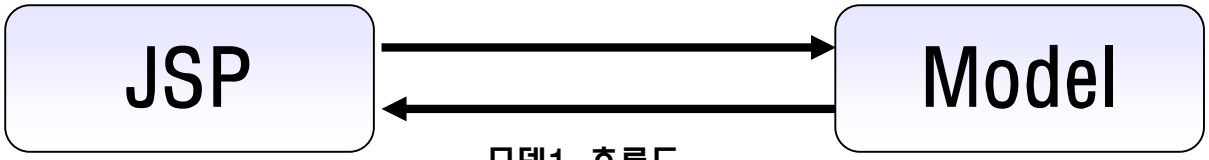


MVC 개발 Framework는 기반 코드를 서블릿으로 작성합니다.
서블릿 코드를 직접 구현하는 일은 드물지만,
반드시 숙지를 하고 있어야 하는 사항입니다.

Model 1

- 모든 클라이언트의 요청과 응답을 JSP가 담당하는 구조입니다.
- 프리젠테이션 로직인 디자인 코드(HTML)와 비즈니스 로직(자바 코드)를 구분하지 않고 하나의 JSP 파일 내에 함께 기술해서 웹 프로그램을 제작하는 방식입니다.
- Controller가 따로 존재하지 않습니다.

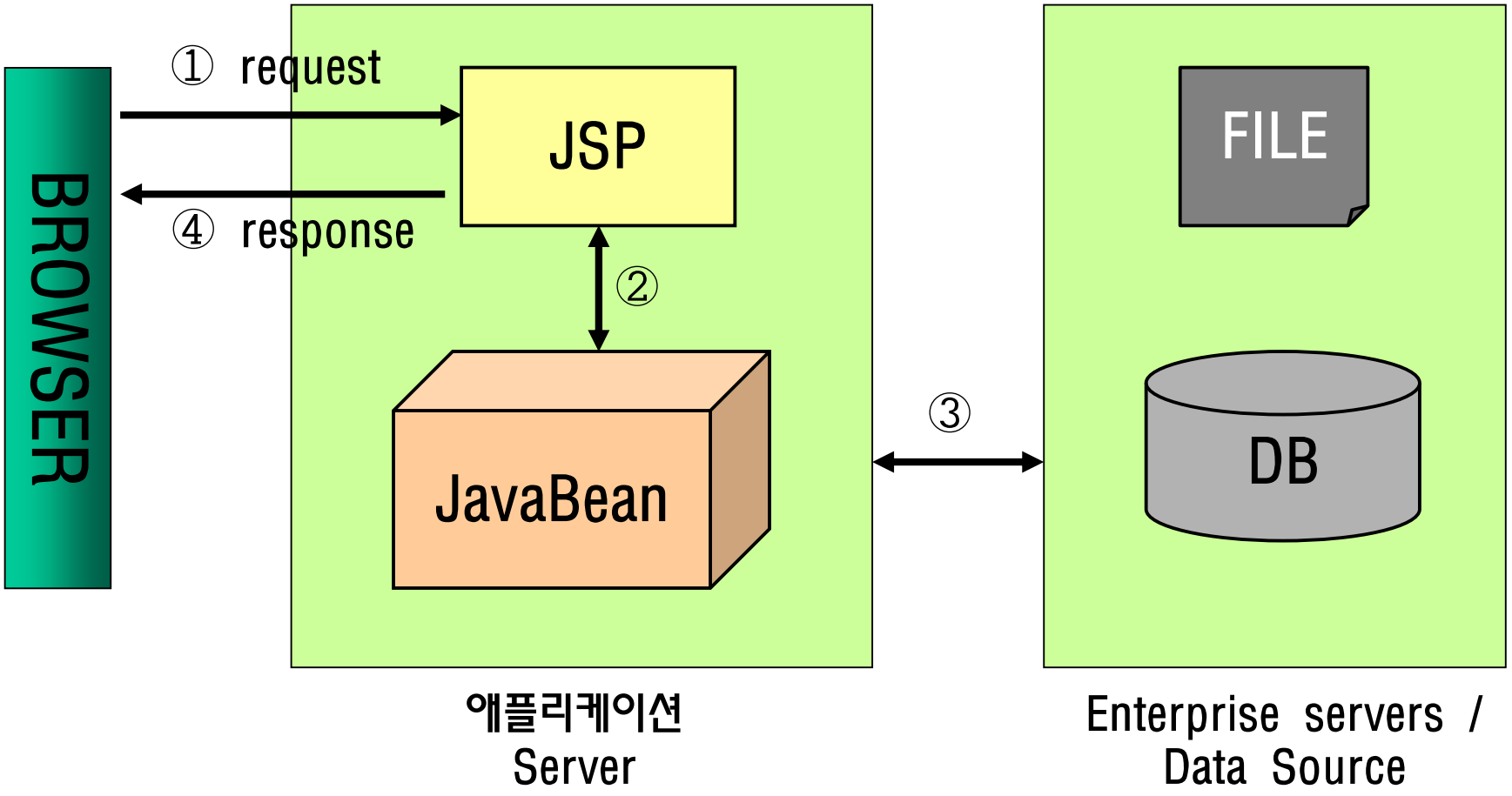
Model1 개발	설명
장점	개발하기 쉽고, 배우기 쉽습니다. 구조가 간단하고 쉽게 배울수 있으며 성능면에서 유리합니다. 중소형 프로젝트에 적합합니다.
단점	복잡한 프로그램 개발에는 부적합합니다. 유지 보수 하기가 어렵고, 이해하기 어렵습니다. 디자인 코드와 비즈니스 로직의 구분이 명확하지 않아 복잡도가 높습니다. 수정 사항이 발생했을 때마다 디자이너와 개발자간의 협업이 필요합니다. 비즈니스 로직의 재사용성이 떨어집니다.



모델1 흐름도

다음 페이지에 계속…

Model 1


① request
④ response
BROWSER
JSP
②
JavaBean
③
FILE
DB
애플리케이션
Server
Enterprise servers /
Data Source

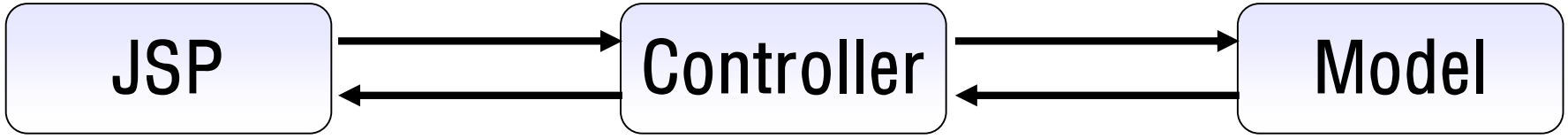
Model2

- 디자인(디자이너)과 로직 부분(개발자)을 나누어 개발(역할 분담)합니다.
- 즉, HTML 페이지 작성과 DB 접근 작업을 분리하여 코딩합니다.
- Controller 역할을 수행하는 Servlet 프로그램의 역할이 중요합니다.
- Model2 = Model + View + Controller + etc
- 프로그램 초기 진입점을 Controller가 담당합니다.

일반적으로 Model2 방식을 MVC 패턴이라고 부릅니다.

구성 요소가 Model, View, Controller으로 구성 되었기 때문입니다.

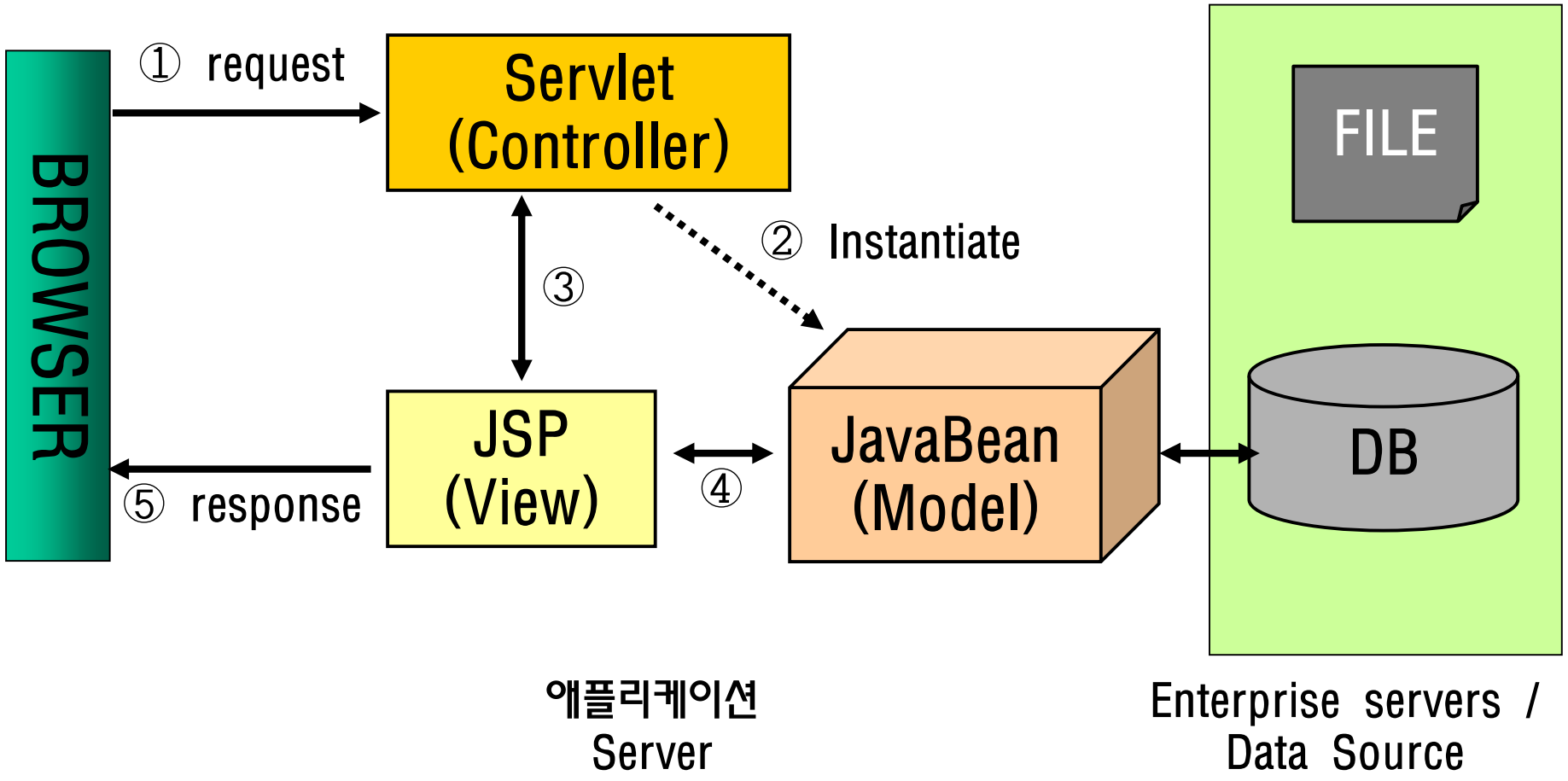
Model2 개발	설명
장점	디자인 코드와 비즈니스 로직이 분리되어, 비즈니스 로직의 재사용성이 좋아집니다. 비즈니스 로직 계층의 확장성이 용이합니다. 처리 작업의 분리로 인해 유지 보수와 확장이 용이합니다. 개발자와 디자이너 간의 역할과 책임 구분이 명확해집니다.
단점	초기 설계에 많은 시간이 소요됩니다. 개발자는 MVC 패턴에 대한 개념(MVC 구조에 대한 이해)이 필수적으로 요구됩니다. 또한 개발자의 높은 Skill이 요구됩니다.



모델2 흐름도

다음 페이지에 계속…

Model2 구조


BROWSER
① request
Servlet
(Controller)
② Instantiate
③
JSP
(View)
④
JavaBean
(Model)
⑤ response
FILE
DB
애플리케이션
Server
Enterprise servers /
Data Source

MVC 패턴의 개요

- MVC = Model + View + Controller
- MVC 각 영역간의 고유한 역할에 집중을 하여 개발하기 위함입니다.
- 다른 영역의 변경에 의한 영향을 최소화하기 위함입니다.

항목	실체	Model1	Model2	Role(역할)
Model	Bean	O	O	로직
View	JSP	O	O	디자인
Controller	Servlet	X	O	제어

MVC 패턴을 위한 최소 Skill

- MVC 패턴을 사용하기 위하여 요구되는 최소한의 Skill은 다음과 같습니다.

항목	내용
HTML 관련	HTML, 자바스크립트, CSS
scripting Elements	표현식, 지시어, 선언문, 스크립트 릿, 내장 객체 등등
액션 태그	RequestDispatcher을 이용한 forward 사용법. include Action tag 사용법
scope에 대한 이해	ServletContext(application), HttpSession(session) , HttpServletRequest(request)
데이터 처리	DAO와 DTO/VO/Bean 등에 대한 이해와 작성법.
서블릿 작성법.	서블릿 mapping, Annotation에 대한 이해가 필요합니다. 서블릿 Filter, Listener에 대한 이해가 필요합니다.
DataSource 사용법	커넥션 풀(Connection Pool)에 대한 이해와 작성법
Command 패턴	사용자의 어떤 요청의 명령어를 보고 이를 처리하는 개발 패턴 요청 파라미터를 이용하여 사용자의 요청을 서블릿으로 넘깁니다.

프레임 워크(Framework)의 등장 배경

- 코드를 재사용한다면 많은 시간과 노력을 절약할 수 있습니다.
- 재사용성 높은 프로그램을 만드는 개발자가 우대되어야 합니다.

동작에 필요한 구조를 어느 정도 완성해 놓은 반제품 형태의 도구로써 완전한 애플리케이션 S/W는 아니다.

Framework(구조, 뼈대)

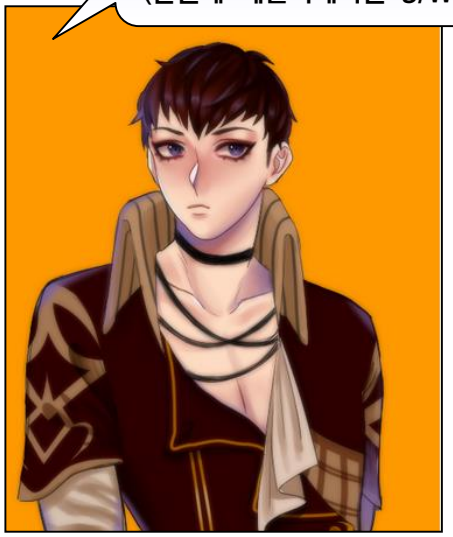
+

뼈대(프레임워크)에 추가 재료로 살을 붙이는 작업이 필요합니다.

보충/첨가할 재료

=

살을 붙임으로써 완전 제품 형태의 도구를 만들어 낸다.
(완전체 애플리케이션 S/W)



최종 결과물

어떤 사람이 작업을 하든 기본적인 구조/뼈대는 동일 합니다.

물감대신, 크레파스, 색연필 등을 사용하여 보충 점가할 수 있습니다.

추구하고자 하는 목적 및 기호에 따라서 완전체인 최종 결과물을 도출합니다.

프레임워크(Framework)

- 사전적인 의미는 [어떤 것을 구성하는 구조, 뼈대]를 의미합니다.
- 어떤 문제를 해결하기 위한 잘 설계된 재사용이 가능한 코드(모듈)들의 집합을 말합니다.
- 다시 말해서 동적인 웹 페이지/웹 애플리케이션을 만들어 내기 위한 유틸리티 모음집.
- 구성 요소
  - 클래스, 인터페이스, 정보 파일(XML)

- 프레임워크의 기대 효과
  - JDBC 프로그래밍의 복잡함이나 번거로움 없이 간단한 작업만으로 데이터베이스와 연동되는 시스템을 빠르게 개발할 수 있습니다.

- 프레임워크의 구분

구분	설명
기능 프레임워크	특정 기능 구현에 사용합니다.
지원 프레임워크	개발을 지원하는 프레임워크 ANT, JUnit 등등
통합 프레임워크	여러 기능들을 프레임워크 한 곳에 통합시킨 프레임워크.

프레임워크의 종류

- JAVA 관련하여 대표적인 프레임워크(Framework)는 다음과 같습니다.

Framework	설명
스트럿츠	struts 1, struts2 MVC 모델2에 기반한 웹 프레임워크 웹 개발을 위한 오픈 소스 웹 애플리케이션 Model 영역 : EJB, iBatis, JDBC 등등 View 영역 : 태그 라이브러리 제공 http://struts.apache.org
스프링	spring2.5, spring3.0, spring4.0
마이바티스	iBatis(아이바티스), MyBatis(마이바티스) SQL 문장으로 직접 DB 데이터를 다루는 SQL 맵퍼(mapper)
하이버네이트	Hibernate 자바 객체를 통해 간접적으로 DB 데이터를 다루는 객체 관계 맵퍼 (Object-Relational mapper)입니다. 탑-링크(TopLink)도 ORM에 속합니다.

재사용 방식

- 재사용 방식은 다음과 같은 항목들이 존재합니다.(Framework)는 다음과 같습니다.

재사용 방식	설명
Copy & Paste 재사용	전통적인 방식으로 [복사 후 붙여 넣기]
함수의 재사용	동일한 코드를 함수로 선언 후 재사용하기
클래스 상속과 재정의를 통한 재사용	객체의 등장은 한 단계 높은 수준의 재활용 방식 지원 추상 객체의 클래스 상속을 통한 공통점의 공유 변경이 필요한 부분만 재정의
디자인 패턴과 프레임 워크	개발자간 공유하는 경험들의 모음(디자인 패턴)들을 모아서 적용하기 쉽도록 묶어둔 집합체. 특정 문제를 위한 클래스와 인터페이스의 집합체.

좋은 프레임워크의 조건

- 재사용 방식은 다음과 같은 항목들이 존재합니다.(Framework)는 다음과 같습니다.

항목	설명
재사용성	한 번 작성된 코드는 다른 영역에서 재사용이 가능해야 합니다.
영속성 (Persistence)	지속성이라고도 합니다. 애플리케이션을 종료 후 다시 시작하면 이전에 저장한 데이터를 다시 불러올 수 있는 기술을 의미합니다.
확장성	예를 들어서 PlugIn 등의 형태로 다른 프로그램에 대하여 확장이 가능해야 합니다.
용이성	사용하기 쉬워야 합니다.
독립성	특정 Vender(밴더)의 플랫폼에 독립적이어야 합니다. 꾸준한 지원(버전 업, 패치 등)이 되어야 합니다.