올인원
스프링 프레임워크

Chapter 01
웹 프로그래밍의 이해

목차

학습목표

- 웹 프로그래밍에 대해 이해합니다.
- 웹 서버와 웹 애플리케이션 서버의 역할과 차이점에 대해 살펴봅니다.
- 웹 프로그래밍 언어 종류와 특징에 대해 살펴봅니다.
- 웹 프로그래밍의 구조에 대해서 살펴봅니다.
- MVC 패턴에 대해 살펴봅니다.
- 스프링 MVC의 구조에 대해 살펴봅니다.

Section 01

웹과 웹 프로그래밍

- **하이퍼텍스트**
  - 영국의 컴퓨터 과학자 팀 버너스리가 고안함
  - 인터넷 공간에 존재하는 문서와 문서를 클릭만으로 쉽게 이동할 수 있는 것을 의미하는데, 이는 월드 와이드 웹의 시초가 됨
- **정적 페이지(static page)**
  - 초창기 웹은 주로 정적 페이지를 이용하는 서비스였음
  - 주로 서버는 클라이언트의 요청에 따라 매번 변화가 없는 정해진 페이지를 반환하는 역할을 함

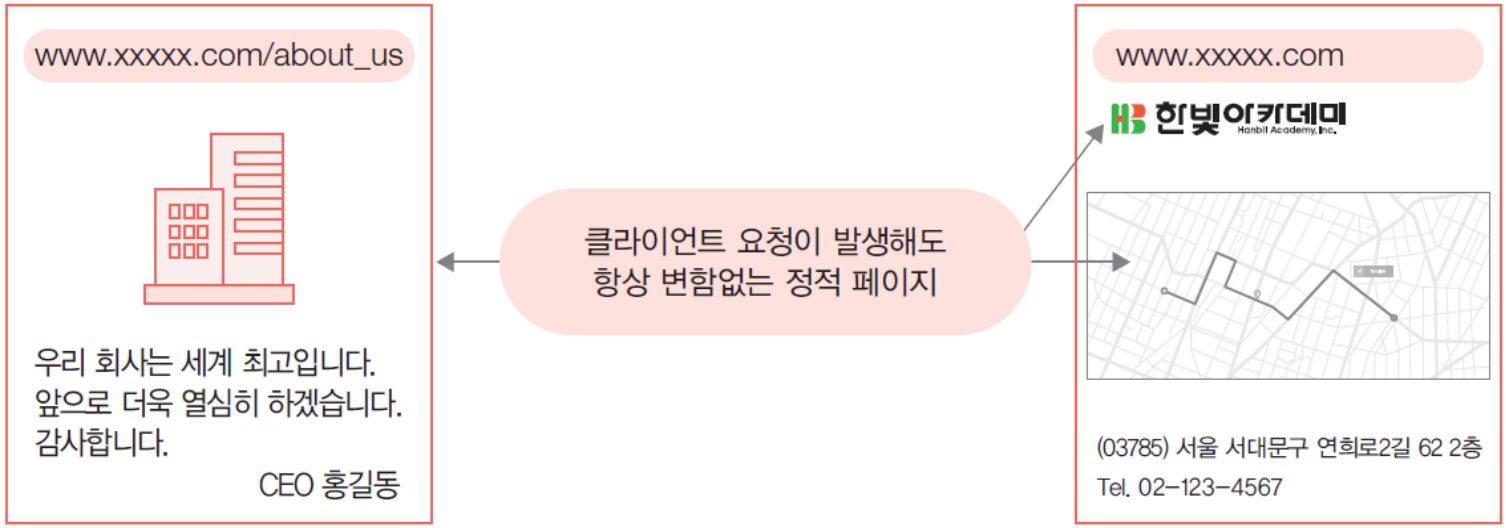

그림 1-1 항상 변함이 없는 정적 페이지

- **동적 페이지(dynamic page)**
  - 구글, 네이버 등에서 이루어지는 검색, 로그인, 신용카드 결제 등은 모두 동적 웹 서비스가 필요한 업무임
  - 이를 구현하기 위해 ASP, PHP, JSP 같은 웹 프로그래밍 언어가 등장함

그림 1-2 요청에 따라 변화하는 동적 페이지

■ 웹 애플리케이션을 만들기 위한 서버 구성

- 웹 서버와 웹 애플리케이션 서버로 구분함

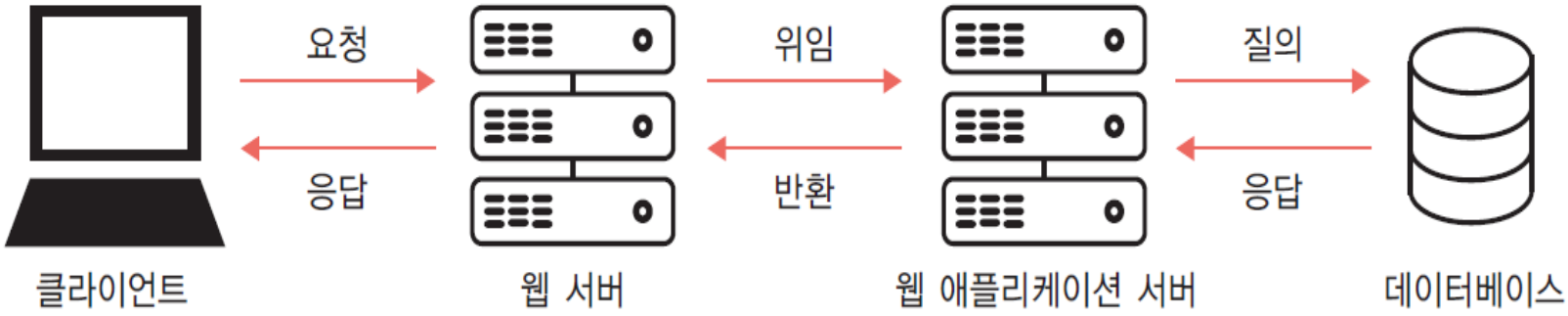

그림 1-3 클라이언트와 서버 구성도

<여기서 잠깐!> 클라이언트에는 어떤 것들이 있나요?

서버에 서비스를 요청하고 서비스를 제공받는 쪽을 클라이언트라고 합니다. 대표적인 클라이언트로는 브라우저가 있습니다. 우리는 브라우저를 이용해서 서버에 정보를 요청하고 서비스를 제공받기 때문입니다. 브라우저 외에 명령 프롬프트, 스마트폰 앱, FTP 프로그램 등도 모두 클라이언트라고 하는데요. 서버에 특정 요청을 보내고 결과를 받는다면 모두 클라이언트라고 생각하면 됩니다.

■ 서버와 클라이언트의 관계

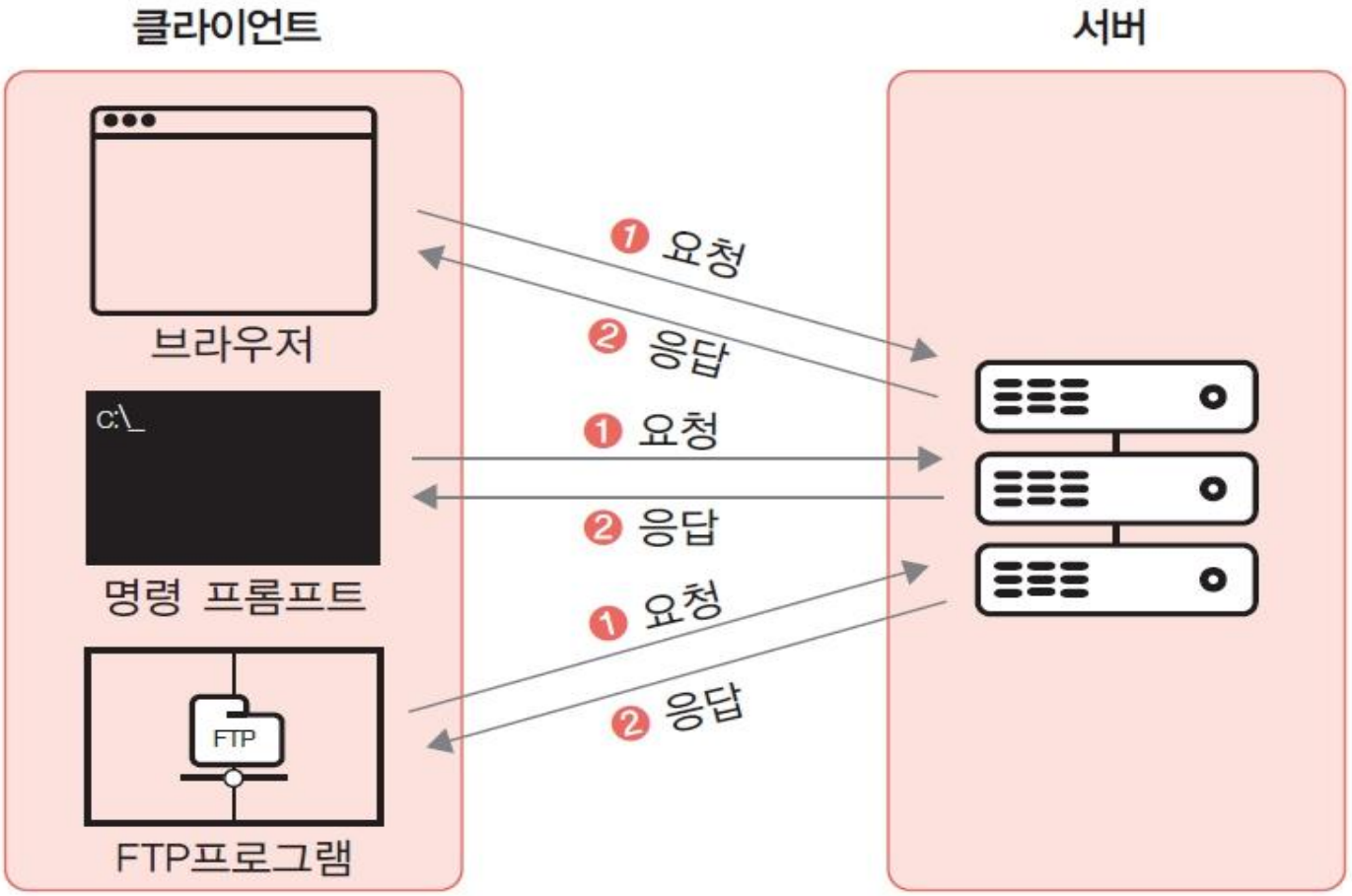

그림 1-4 서버와 클라이언트의 관계

웹 서버

- 클라이언트의 요청을 가장 먼저 받는 서버로 클라이언트와 직접 통신함
- 웹 서버의 주요 역할
  - ✓ 클라이언트의 요청에 따른 정적 페이지 응답
  - ✓ 웹 서버에서 직접 응답하는 대표적인 콘텐츠: 웹 페이지(예 html, css, js), 이미지(예 png, jpg), 음악(예 mp3) 등
  - ✓ 정적 콘텐츠는 웹 서버에 저장되어 서비스됨
  - ✓ 웹 서버는 클라이언트에서 정적 페이지 요청이 아닌 동적 페이지 요청이 있는 경우 웹 서버는 요청을 웹 애플리케이션 서버에 위임함

■ 웹 서버의 주요 역할

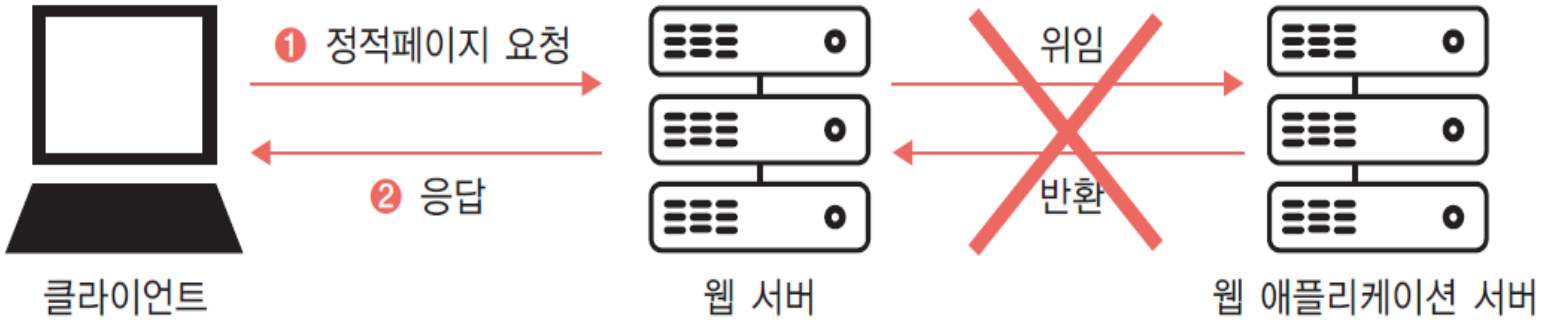

그림 1-5 웹 서버에서 즉시 처리되는 정적 페이지 요청

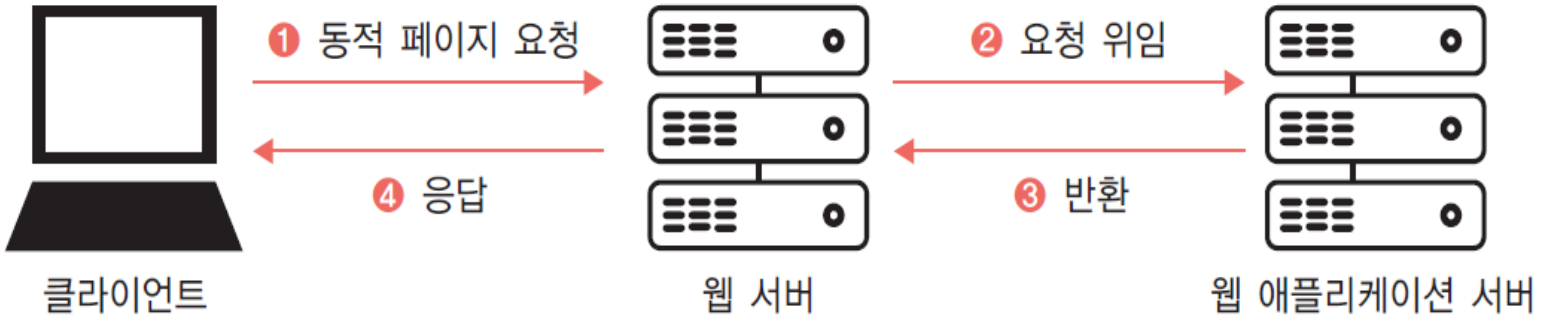

그림 1-6 웹 애플리케이션 서버에 위임되는 동적 페이지 요청

2. 웹 서버와 웹 애플리케이션 서버

<여기서 잠깐!> 웹 애플리케이션 서버가 웹 서버의 역할까지 할 수 없나요?

가능합니다. 웹 서버는 정적인 콘텐츠만을 저장하고 있다가 클라이언트의 요청이 왔을 때 응답하는 비교적 간단한 역할을 하는데, 이러한 역할까지도 웹 애플리케이션 서버가 할 수는 있습니다. 그러나 일반적으로 웹 서버와 웹 애플리케이션 서버를 구분하는 이유는 동적인 업무를 처리하는 웹 애플리케이션 서버의 부담을 최대한 덜어주기 위함입니다.

웹 애플리케이션 서버는 데이터베이스 통신, 외부 네트워크 연결, 외부 인증 같은 스트레스(서버 부하)가 증가할 수 있는 업무가 대부분이기 때문에 웹 서버와 웹 애플리케이션 서버를 분리해서 구성하는 것이 바람직합니다.

표 1-1 웹 서버와 웹 애플리케이션 서버

웹 서버	웹 애플리케이션 서버
Apache Server, IIS, Nginx	Tomcat, JBoss, Jeus

3. 웹 프로그래밍 언어의 종류와 특징

- **ASP**
  - 윈도우의 IIS 웹 서버에서 실행될 수 있는 웹 프로그래밍 언어를 말함
  - VBScript를 이용한 비교적 쉬운 스크립트 언어
  - 과거에는 많은 인기를 얻었지만 최근에는 거의 사용하지 않으며, ASP를 만든 마이크로소프트사(MS)에서도 더 이상 지원하지 않음
- **PHP**
  - C를 기반으로 만들어진 웹 프로그래밍 언어로 빠른 실행 속도와 입문자가 배우기 쉬움
  - 대규모 시스템을 개발하기에 적합하지 않다는 단점이 있음
  - ASP와 달리 지속적인 지원이 이루어지고 있으며, 최근에는 PHP 8.x로 업그레이드됨
  - 특히 소규모 온라인 쇼핑몰에 많이 적용되어 있음

JSP(Java Server Page)

- 자바 Java 기반으로 만들어졌으며 컴파일 이후 서블릿(Servlet)으로 변환되어 서버에서 동적으로 작동함
- 서블릿으로 변환된 JSP는 사용자의 요청을 스레드(Thread)를 이용해 처리하기 때문에 서버 부하 걱정 없이 효율적인 웹 서비스를 제공할 수 있음

<여기서 잠깐!> 스프링 MVC를 학습하기 위해서는 어떤 선행 학습이 필요할까요?

스프링 MVC를 학습하기 위해서는 기본적으로 HTML, CSS, 자바스크립트와 서블릿, JSP에 대한 선행학습이 필요합니다. HTML, CSS, 자바스크립트는 아주 뛰어날 필요는 없고 각각의 언어에 대해 약간의 지식만 가지고 있으면 됩니다.

반면 서블릿, JSP에 대해서는 어느 정도 전문 지식을 가지고 있으면 좋습니다. 스프링 MVC가 서블릿, JSP로 구현하는 웹 프로그래밍을 보다 효율적으로 개발할 수 있게 만들어 주는 프레임워크이기 때문입니다.

<여기서 잠깐!> 퍼블리셔, 프런트엔드, 백엔드가 무엇인가요?(1)

- **퍼블리셔**: 웹문서의 기본 뼈대를 제작합니다. 대체로 HTML, CSS, 자바스크립트를 이용해서 웹문서를 제작합니다. 퍼블리셔는 디자이너와 프런트엔드를 이어주는 중간 다리 역할을 합니다. 더불어 웹 사이트의 전체적인 레이아웃의 제작과 폰트 등을 설정하고 사용자의 편의성을 고려한 버튼, 데이터 입력 양식, 경고 창 등을 제작합니다.

- **프런트엔드**: **퍼블리셔가 제작한 웹 문서에 자바스크립트를 이용해서 동적 페이지를 제작합니다.** 주로 자바스크립트 기반의 라이브러리 또는 프레임워크를 이용하는데, jQuery, Vue.js, ReactJS 등을 주로 이용하며 최근에는 ReactJS의 인기가 많습니다. 프런트엔드 개발자는 페이지를 연결하는 방법과 데이터 전달 방법을 설계합니다. 또한 서버와 통신해서 데이터를 서버로 보내고 반대로 서버의 데이터를 가져오는 기능도 구현합니다.

<여기서 잠깐!> 퍼블리셔, 프런트엔드, 백엔드가 무엇인가요?(2)

- **백엔드**: **웹 서비스에서 클라이언트의 요청을 처리하는 서버의 기능을 구현합니다.** 클라이언트의 요청에 처리하기 위해서 데이터베이스에 접속하기도 하고 때로는 다른 네트워크의 서버와 통신하기도 합니다. 클라이언트와 통신할 때 사용되는 데이터는 주로 JSON을 사용하며, 데이터베이스는 MariaDB, MySQL, MSSQL 등을 사용합니다. 백엔드 개발자는 어떤 프레임워크를 사용하는지에 따라 사용하는 언어가 결정됩니다.

Section 02

model1과 model2

■ model1과 model2

- **웹 애플리케이션을 구현하기 위한 프로그램 설계 방법**
- model1 설계 방법보다 효율적이고 세련된 방법이 model2 설계 방법임
  - ✓ **효율적인 방법이란?**

    개발 시점보다는 유지 보수 측면에서 업무의 효율성을 높일 수 있다는 의미함
  - ✓ **세련된 방법이란?**

    개발 시점에서 각각의 기능을 분리(모듈화)하는 방법을 의미함

2. model1 아키텍쳐

■ model1 설계 방법

- JSP와 자바빈즈(JavaBeans)를 이용해서 웹 애플리케이션을 개발하는 방법

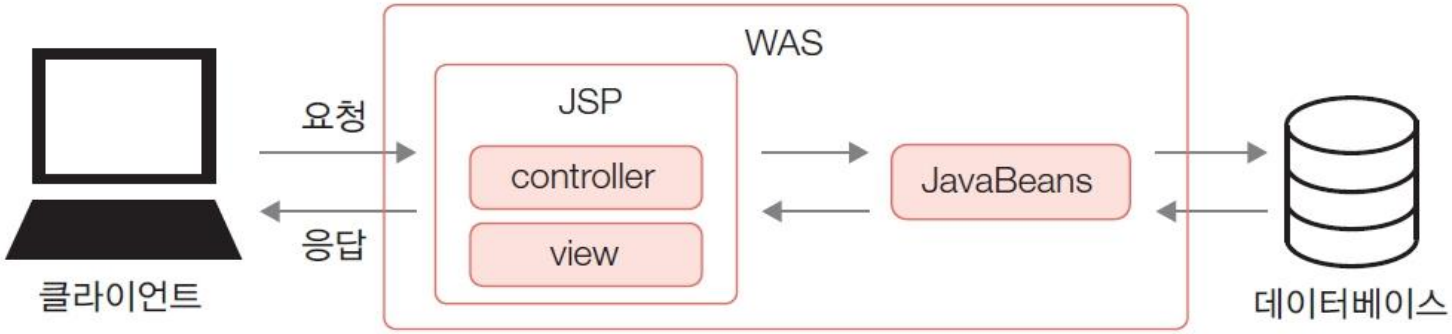

그림 1-7 JSP와 JavaBeans를 이용하는 model1의 구조

- 장점
  - ✓ model1은 다양한 파일을 만들지 않고도 HTML 기반의 JSP와 자바빈즈만을 만들어 개발하기 때문에 개발 속도가 비교적 빠른 편임
  - ✓ 개발 초기에는 개발자의 스트레스를 줄일 수 있음
- 단점
  - ✓ model1은 컨트롤러와 뷰 코드가 JSP에 섞여 있어 유지 보수가 어려움
  - ✓ 특히 대형 프로젝트 개발 및 유지 보수에 어려움이 발생할 수 있음
  - ✓ model1은 과거에 많이 사용한 방법으로 주로 소규모 프로젝트 또는 초급 웹 개발자들이 선호하는 방식이라 최근에는 거의 사용하지 않음

model2 설계 방법

- model2는 mode1의 업그레이드된 버전으로, 각각의 기능을 모듈화하여 기능에 따른 코드를 명확하게 분리하는 방법임
- 웹 애플리케이션을 구현하는 데 가장 많이 사용되는 방법으로, 클라이언트의 요청을 처리하는 컨트롤러와 클라이언트에 응답하기 위한 뷰를 분리해서 유지 보수가 쉬움
- 결과적으로 요청을 받는 것과 전체적인 프로그램의 제어는 컨트롤러가 하고, 비즈니스 로직은 빈이 처리하며, 화면에 출력하기 위한 응답은 뷰가 담당함
- model2는 웹 애플리케이션을 구현하는 데 가장 많이 사용되는 방법으로 MVC 패턴 구조를 따름

■ model2 설계 방법

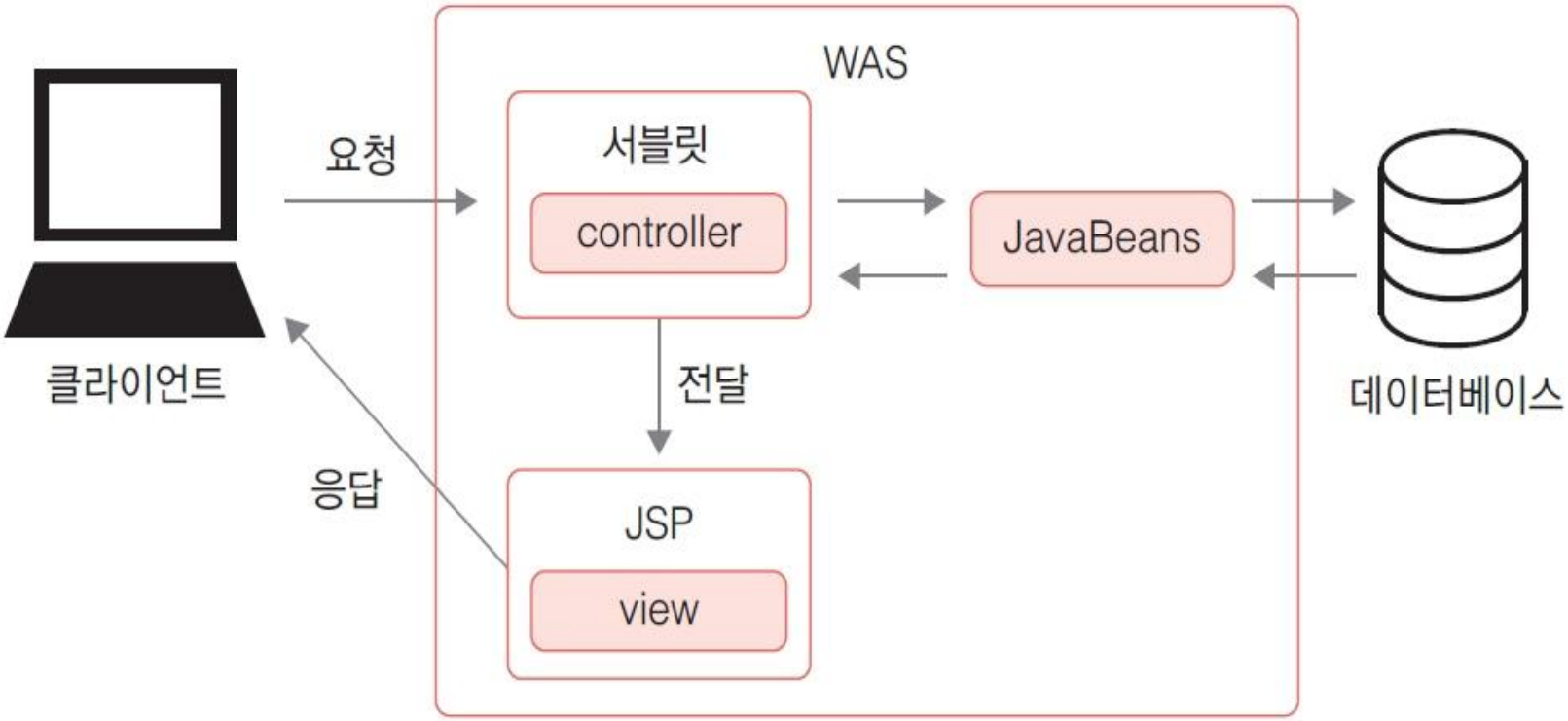

그림 1-8 서블릿, JSP, JavaBeans를 이용하는 model2의 구조

<여기서 잠깐!> 비즈니스 로직이란?

- **비즈니스 로직**: 클라이언트의 특정 요청에 대해서 웹 애플리케이션이 처리해야 하는 업무를 말합니다. 우리가 자주 이용하는 이메일 송수신, 신용카드 결제, 회원가입 및 로그인, 캘린더의 일정 관리, 학사 관리 등 웹 애플리케이션에서 이루어지는 내부 업무가 모두 비즈니스 로직이라고 생각하면 됩니다.

- **프레젠테이션 로직**: 클라이언트 화면에 비즈니스 로직의 결과를 보여주기 위한 화면(디자인) 구성입니다. 로그인할 때 성공 또는 실패 화면이 이에 해당됩니다. model2에서 프레젠테이션 로직은 뷰가 담당하고, 비즈니스 로직은 빈(Java Beans)이 담당합니다.

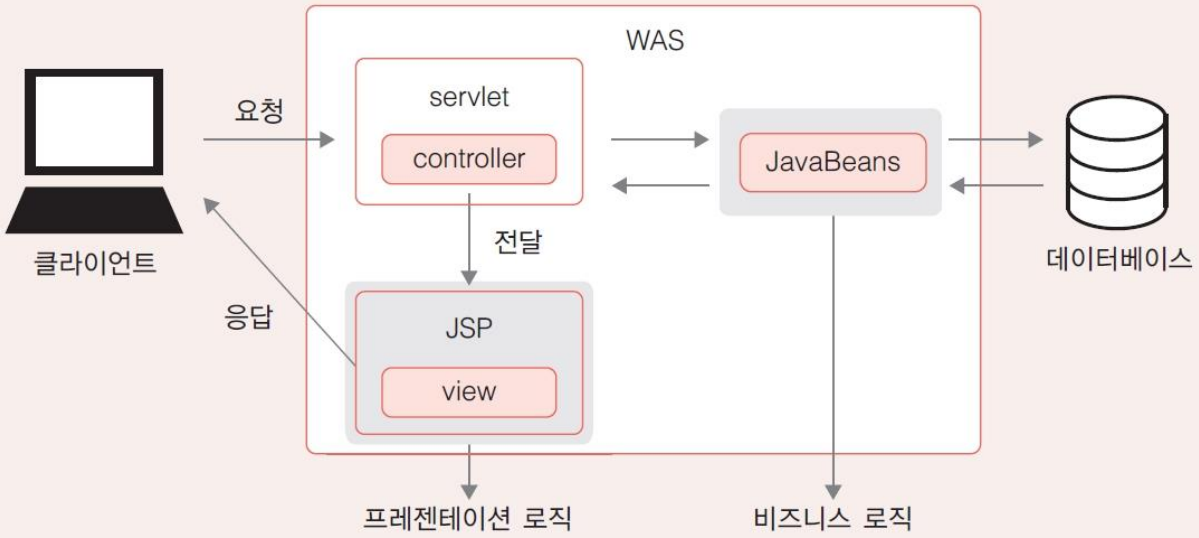

그림 1-9 프레젠테이션과 비즈니스 로직

Section 03

MVC 패턴 이해하기

MVC 디자인 패턴

- 모델, 뷰, 컨트롤러를 이용해서 프로그래밍하는 소프트웨어 설계 방법
- 모델(Model): 데이터베이스와 밀접한 관계를 갖고 비즈니스 로직을 담당함
- 뷰(View)는 클라이언트와 밀접한 관계를 갖고 비즈니스 로직의 결과를 출력하기 위한 화면 구성을 담당함
- 컨트롤러(Controller): 클라이언트의 요청에 대해 모델과 뷰를 컨트롤함

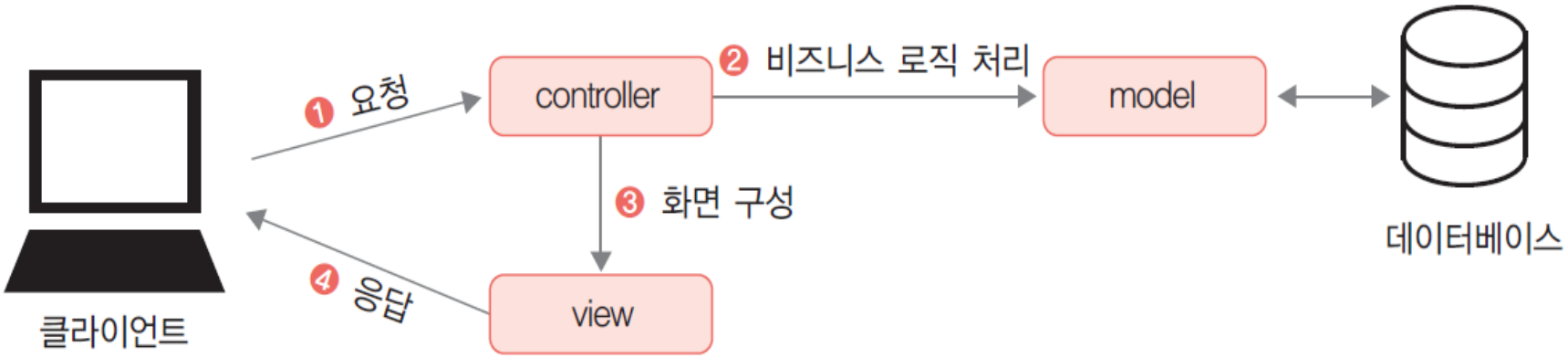

그림 1-10 각각 업무를 분업화 하는 MVC 패턴

MVC 디자인 패턴

- MVC는 웹 프로그래밍에만 사용되는 것은 아니라 프로그램 전반에 사용됨
- MVC를 보완한 MVVM(모델-뷰-뷰 모델), MVP(모델-뷰-프레젠터) 등도 존재함
- mode1, model2는 MVC 패턴을 웹 프로그래밍에 적용한 것임
  - ✓ 최근에 가장 많이 사용되는 웹 프로그래밍 설계 방법은 MVC 패턴에 기반한 model2임
- MVC 패턴의 장점
  - ✓ 프레젠테이션 로직과 비즈니스 로직을 분리할 수 있고, 각각의 기능을 모듈화해서 개발을 체계적으로 진행할 수 있음
  - ✓ 유지 보수 시점에서도 MVC 패턴의 구조로 구축되어 있기 때문에 새로운 개발자는 모델, 뷰, 컨트롤러 각각의 모듈을 찾아가며 전체적인 구조를 파악하기 수월함

<여기서 잠깐!> 디자인 패턴이란?

소프트웨어 설계적인 측면에서 디자인 패턴이란 각기 다른 프로그램이라도 전체적인 설계는 어떤 특정한 패턴을 갖는다는 의미입니다. 그리고 이러한 패턴을 정리해서 문서화한 것을 디자인 패턴이라고 합니다.

디자인 패턴의 종류는 다양한데 가장 유명한 디자인 패턴은 GoF(Gang of Fout) 로, 20여 가지의 패턴을 생성(creational), 구조(structural), 행위(behavioral)로 분류합니다.

■ 스프링 MVC

- 웹 애플리케이션을 개발하기 위한 전용 프레임워크
- 기본적으로는 MVC 패턴을 따르기 때문에 model2와 전체적인 구조는 비슷함
  - ✓ 다시 말해 모델, 뷰, 컨트롤러를 스프링에 적용해서 웹 애플리케이션 개발을 더욱 쉽고 체계적으로 진행할 수 있도록 고안된 프레임워크임
  - ✓ 따라서 스프링 MVC 구조를 학습할 때 머릿속에 MVC 패턴을 그려놓고 학습하면 쉽게 이해할 수 있음

■ 스프링 MVC의 흐름

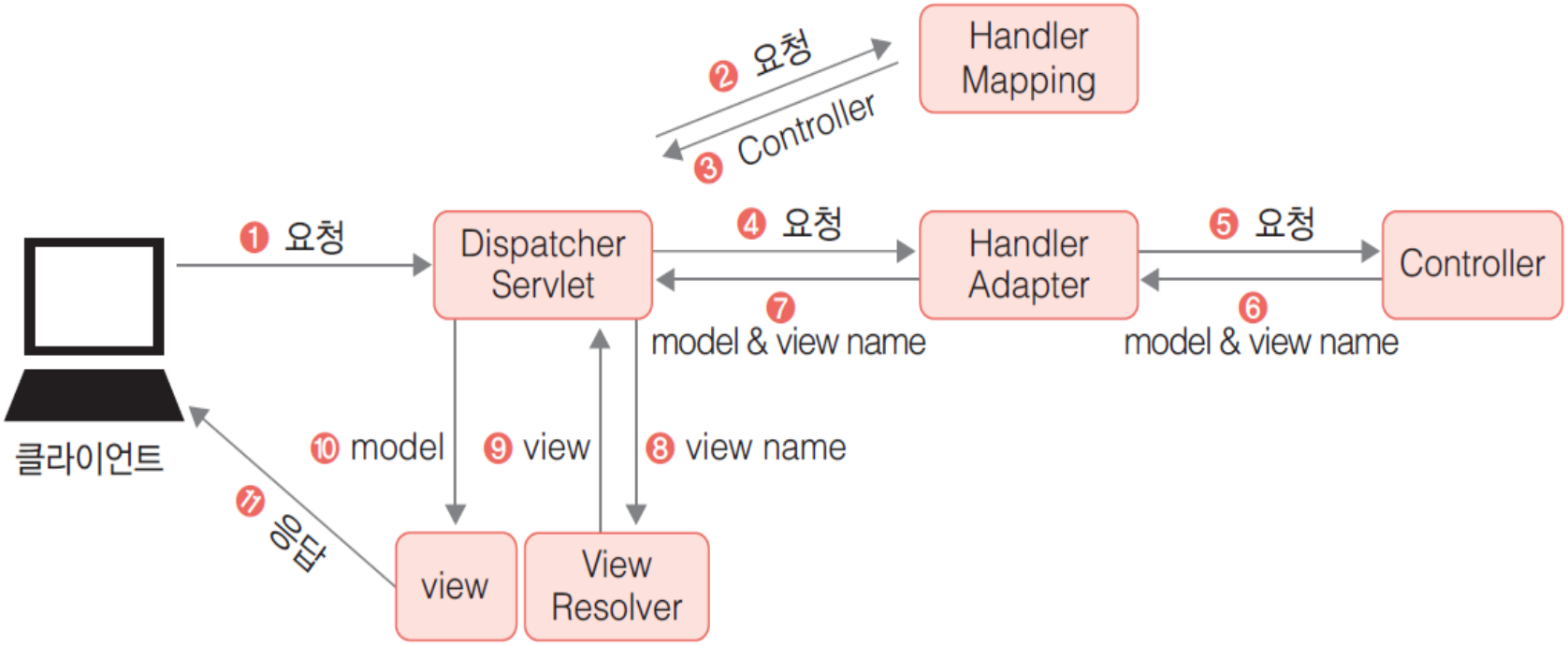

그림 1-11 스프링 MVC 프레임워크의 주요 모듈

스프링 MVC의 흐름

- 1단계
  - ✓ 클라이언트로부터 요청이 들어오면 DispatcherServlet, HandlerMapping, HandlerAdapter를 이용해 클라이언트 요청에 적합한 컨트롤러를 찾고 해당 컨트롤러의 메서드를 실행함
- 2단계
  - ✓ DispatcherServlet이 클라이언트의 요청을 처리한 결과를 HandlerAdapter로부터 받으면, 처리 결과를 클라이언트에 응답하기 위해 뷰를 찾는 단계
- 3단계
  - ✓ 클라이언트의 요청에 응답하기 위한 마지막 단계
  - ✓ DispatcherServlet 객체는 ViewResolver 객체가 보내준 뷰 정보를 이용해 뷰 객체를 준비함
  - ✓ 뷰 객체는 일반적으로 JSP 파일이며, JSP 파일은 WAS를 통해 클라이언트의 브라우저에 응답 결과를 출력하고 모든 작업을 종료함

■ 스프링 MVC 프레임워크를 사용하는 이유

- 스프링 MVC 프레임워크를 이용하는 것이 model2 설계 방법보다 복잡하게 느껴질 수도 있음
- 하지만 [그림 1-11]에서 살펴본 다양한 객체들 중 개발자가 실제로 코딩해야 하는 객체는 컨트롤러와 뷰뿐임
- 다시 말해 DispatcherServlet, HandlerMapping, HandlerAdapter 객체는 스프링 MVC 프레임워크에 이미 만들어져 있음
- 따라서 개발자는 프레임워크의 구조를 이해하는 수준에서 개발을 진행하면 되고 직접 구조를 만들지 않아도 됨
  - ✓ 개발자는 스프링MVC 프레임워크라는 틀 안에서 컨트롤러와 뷰를 제작하는 작업만 집중하면 됨