MVC 디자인 패턴

MVC 관련 용어

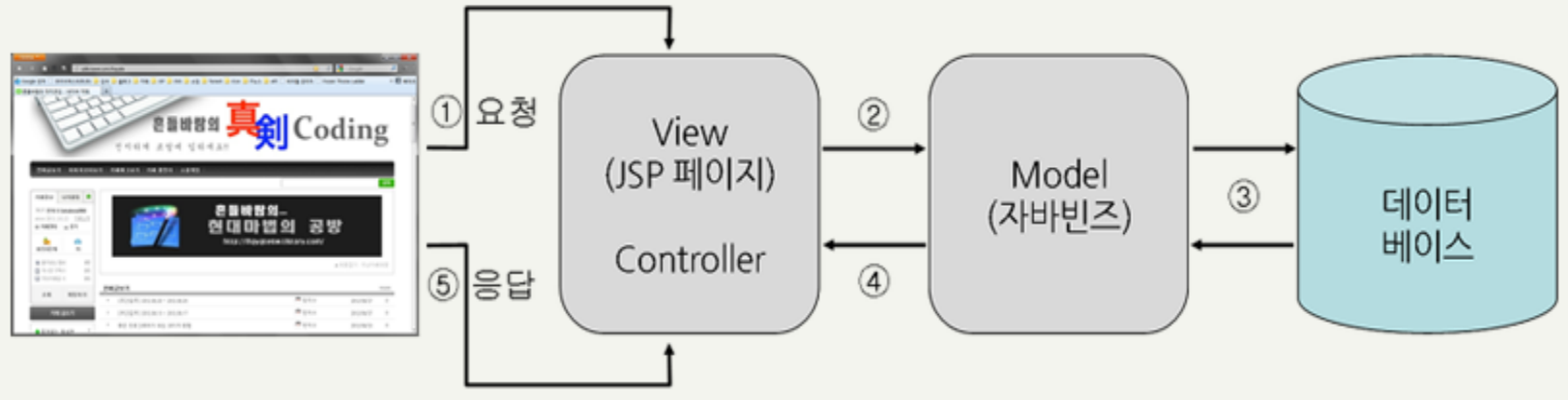
용 어	설 명
Model	- 정보를 담고 있는 객체 - JAVA 클래스(Java Bean)
View	- 정보를 표현하는 객체 - JSP(Java Server Page) 구현
Controller	- 정보 컨트롤, Model과 View 매개체 역할의 객체 - JAVA 클래스(Servlet)

※ 모델2는 데이터(Model) , 표현(View), 정보처리(Controller)를 구분하여 재사용과 가독성을 높이려는 기법이다.

MVC 디자인 패턴

● 모델 1 구조

- JSP로 구현한 기존 웹 어플리케이션은 모델 1 구조로 웹 브라우저의 요청을 JSP 페이지가 받아서 처리하는 구조

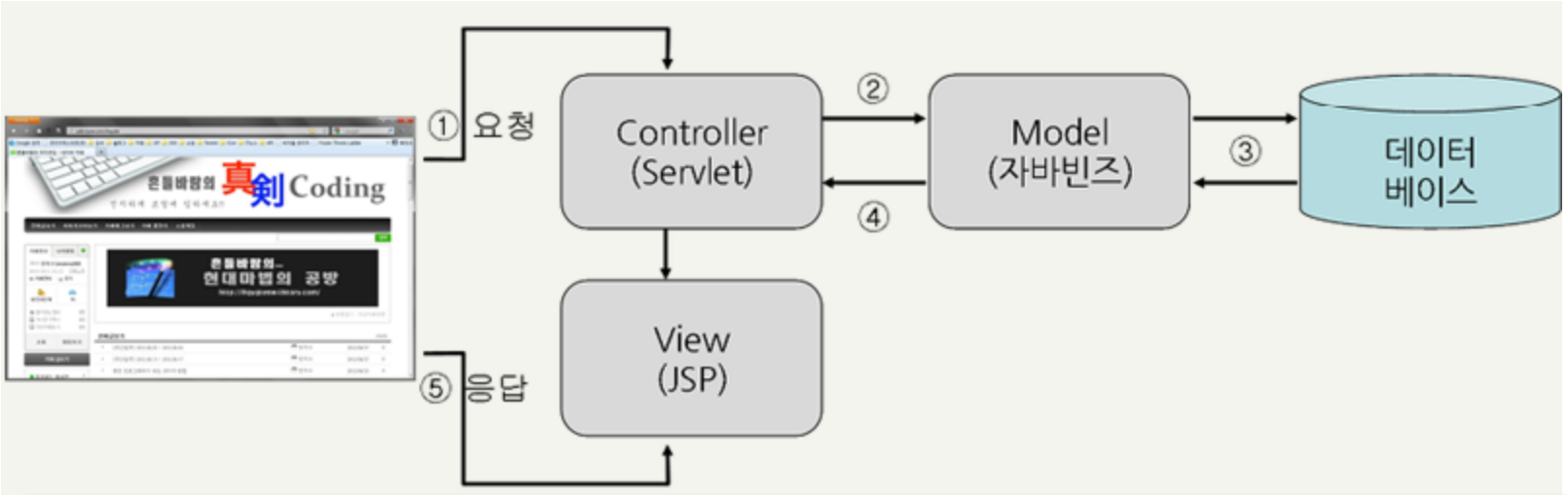


- JSP 페이지 : 비즈니스 로직 처리 코드와 웹 브라우저에 결과를 보여주기 위한 출력 관련 코드가 뒤섞여 있는 구조
- JSP 페이지 안에서 모든 정보를 표현(View)하고 저장(Model)하고 처리(Control)되므로 재사용이 힘들고, 가독성 떨어짐

MVC 디자인 패턴

● 모델 2구조

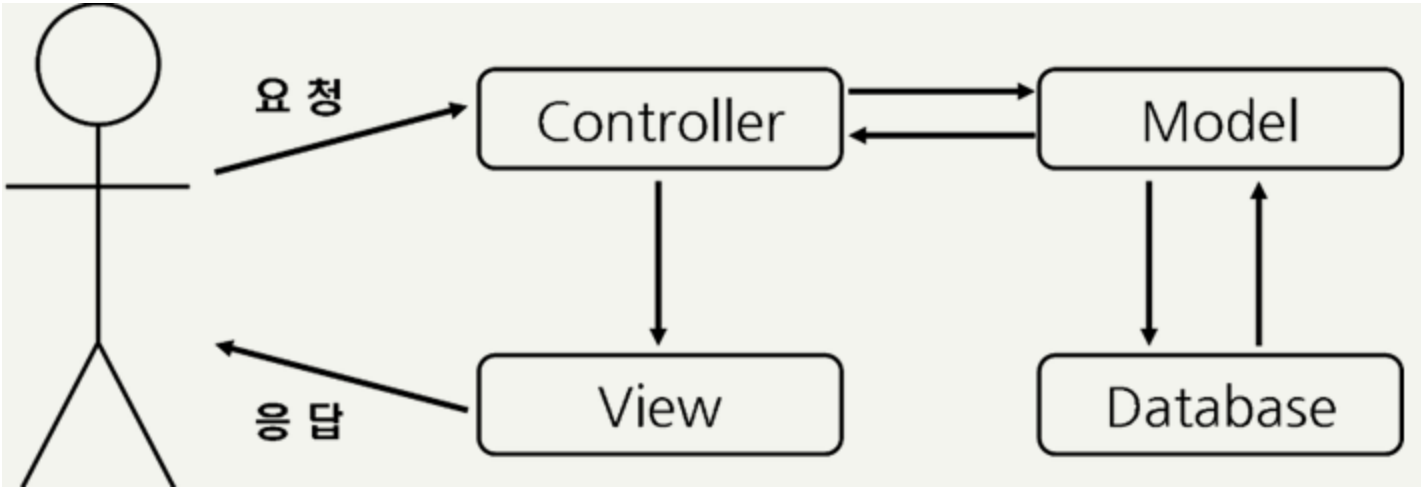
- 모델 2로 구현한 JSP는 데이터 저장을 Java Beans, 데이터처리는 Servlet, 표현은 JSP
- 비즈니스 로직과 데이터, 표현을 완전히 분리



▪ 모델 2 장점
✓데이터 처리나 비즈니스 로직만을 부분적으로 바꿔도 전체에는 영향이 없을 뿐만 아니라 다양한 확장 기능과 유지보수 문제 해결

MVC 디자인 패턴

- 프로그램에서 가장 중요한 것이 비즈니스 로직
- 데이터를 어떻게 처리할까? -> 컨트롤러(Controller)의 역할).
- 데이터베이스에 접근하는 것은 모델(Model) 객체의 몫
- 뷰(View)는 화면 UI를 구성하는 요소, 웹 어플리케이션에서는 JSP가 뷰의 역할



MVC 디자인 패턴의 UML 모델링

MVC 디자인 패턴의 핵심

① MVC 패턴의 가장 큰 장점은 모델과 View의 분리

- ✓ 화면 UI를 위한 코드와 비즈니스 로직을 위한 코드가 섞이지 않는다는 것
- ✓ 화면 UI 개발자와 비즈니스 로직 개발자 분리 가능

② 모델과 View가 분리되면 한 모델에 여러 가지 다양한 View를 붙이는 것도 가능

③ 요구사항의 복잡도는 높지만 규격이 잘 정해진 애플리케이션을 만들 때는 개발자의 일을 획기적으로 줄일 수 있다.

④ 모델과 View의 분리를 효과적으로 할 수 있는 컨트롤러를 제공

- **MVC 패턴의 Controller 역할 ➔ Servlet**
  - ① 웹 브라우저(클라이언트)의 요청을 받는다.
  - ② 웹 브라우저(클라이언트)가 어떤 기능을 요청했는지 분석한다.
  - ③ 해당 비즈니스 로직을 처리하는 Model 호출
  - ④ Model로부터 전달받은 결과물을 알맞게 가공(DB 저장, 삭제, 수정)
  - ⑤ 웹 브라우저(클라이언트)에 처리결과를 보여주기 위한 JSP를 선택한 후 해당 JSP 포워딩

- **MVC 패턴의 View 역할 ➔ JSP**
  - View(요청한 결과를 보여주는 프레젠테이션)의 역할은 JSP 구현
  - MVC패턴에서 View의 역할을 하는 JSP는 비즈니스 로직과 관련된 코드가 없다는 점을 제외하고는 일반 JSP 페이지와 동일
  - request나 session 기본 객체에 저장한 데이터를 사용해서 알맞은 결과를 출력

MVC 패턴의 Model 역할 ➔ Java Beans(DAO)

- Model 기능 : 웹 브라우저의 요청을 **처리**하는데 필요한 기능 제공
- 컨트롤러 요청 ➔ Model 호출 ➔ Model 실행
- 컨트롤러가 요청한 작업을 처리한 후 알맞은 결과를 컨트롤러에게 전달
  - 처리한 결과값은 Java Beans에 저장
- Model 수행 절차

  ① 컨트롤러로부터 요청을 받는다.

  ② 비즈니스 로직을 수행한다.

  ③ 수행결과를 컨트롤러에 리턴한다.