제12장 Spring MVC 개념 2023기07 하태정

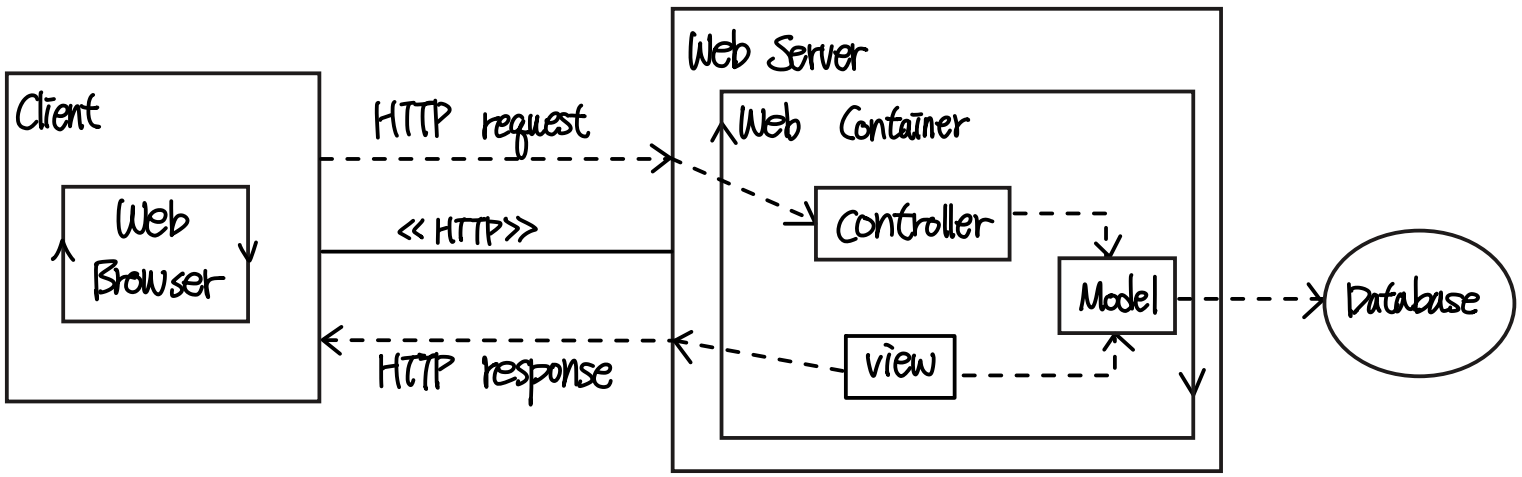
1. MVC 모델의 기초 개념
   - 정의: 비즈니스 로직과 표현(레이어)을 분리하여 웹 응용 프로그램을 개발하고자 하는 디자인 방안입니다.
     애플리케이션을 구성하는 패턴 중 하나입니다.
   - 구성 요소: M은 Model, V는 View, C는 Controller를 의미합니다.
   - 분리의 장점:
     - 개발 효율성: 디자이너는 표현에, 프로그래머는 비즈니스 로직에 전념하여 개발 효율성이 높아집니다.
     - 유지 보수: 웹 응용프로그램의 수정, 확장, 유지보수가 용이합니다.

2. MVC 구성 요소별 역할 및 구현
   - Model (모델)
     - 역할: 사용자 비즈니스 로직 처리. 데이터의 상태 변화가 있을 때 컨트롤러와 뷰에 통보
     - 구현 프로그램: 자바빈즈

   - View (뷰)
     - 역할: 표현 부분 처리. 사용자에게 보여질 화면을 생성하기 위해 모델로부터 정보를 얻어 옴
     - 구현 프로그램: JSP, HTML

   - Controller (컨트롤러)
     - 역할: 적절한 Model을 처리하여 뷰로 제어 이동. 모델에 명령을 보내 모델의 상태를 변경하고,
       데이터(Model)와 화면 요소(View)를 연결 및 관리
     - 구현 프로그램: 서블릿, JSP

3. MVC 모델 시스템 패턴



제12장 Spring MVC 개념

4. MVC 모델의 발전 과정 (V1 → V4 요약)

◦ MVC V1 (서블릿 기반)

- 구조 특징: HttpServlet을 직접 사용하여 요청/응답 처리
- 의존성 문제점: 모든 로직이 하나의 클래스에 집중되어 코드 중복 발생
  Controller, View, Model 역할 구분이 불명확.
- Spring MVC 유사성: 초기 버전

◦ MVC V2 (Controller + View 분리)

- 구조 특징: Controller 인터페이스 및 View 객체 (MyView) 도입으로 공통 구조 확립.
- 의존성 문제점: request/response에 여전히 의존
- Spring MVC 유사성: 구조 명확, 코드 재사용성 증가.

◦ MVC V3 (FrontController + ModelView)

- 구조 특징: ModelView 객체 도입, HttpServletRequest/Response 의존 제거.
  FrontController가 중앙에서 모든 요청 처리
- 의존성 문제점: 컨트롤러 호출 시 파라미터 매핑 수동 처리 필요
- Spring MVC 유사성: ViewResolver로 논리적 View 이름 분리.

◦ MVC V4 (단순화된 실무형 구조)

- 구조 특징: Controller가 Model(Map)과 View 이름만 반환
  FrontController가 Model 생성 및 ViewResolver 자동 호출
- Spring MVC 유사성: Spring의 @Controller, Model, ViewResolver 구조의 기반 완성형

제12장 Spring MVC 개념

5. Spring MVC의 핵심 구성 요소

5.1 DispatcherServlet의 이해

- 개념: Spring MVC의 핵심 Front Controller 패턴 구현체
- 역할: 모든 클라이언트 요청을 가장 먼저 받아 컨트롤러(Handler)로 전달하고, 결과를 뷰(View)에 연결해주는 역할 (요청 → 컨트롤러 → 뷰 흐름을 총괄)
- 주요 동작 과정
  1. DispatcherServlet이 요청을 수신
  2. HandlerMapping을 통해 어떤 @Controller 메서드가 실행될지 결정.
  3. HandlerAdapter가 해당 메서드를 실행
  4. 반환 결과(ModelAndView)를 받아 ViewResolver로 뷰 선택
  5. 선택된 View에 Model 데이터를 전달하여 최종 HTML 렌더링.

5.2 @Controller

- 역할: Spring MVC에서 요청을 처리하는 클래스임을 표시하는 어노테이션
- 특징: 메서드 단위로 @GetMapping, @PostMapping 등을 사용해 요청을 매핑하여 특정 URL 요청을 처리합니다.
- 데이터 처리: 요청을 처리하고 Model에 데이터를 저장한 후, 뷰 이름을 반환하여 DispatcherServlet이 알맞은 뷰를 찾아 렌더링하게 합니다.
- @RestController와의 차이
  - @Controller : 뷰 반환 중심 (JSP, Thymeleaf 등)
  - @RestController: JSON/XML 등 데이터 자체 반환 중심 (REST API)

5.3 Model 인터페이스 (Model / ModelMap / ModelAndView)

- Model의 기본 개념: 뷰에서 출력해야 할 데이터를 담는 그릇(Container)이며, 컨트롤러가 처리한 데이터를 뷰로 전달하는 매개체입니다.
- 데이터 추가: model.addAttribute(String key, Object value) 메서드를 사용해 뷰에서 사용할 데이터를 등록합니다.

제12장 Spring MVC 개념

5. Spring MVC의 핵심 구성 요소

5.3 Model 인터페이스 (Model / ModelMap / ModelAndView)

- 유사 객체 비교
  - Model
    - 주요 역할 : 컨트롤러에서 뷰로 데이터 전달용
    - 상속 관계 : 인터페이스
    - 뷰 지정 방식 : View 이름을 리턴 값으로 반환
    - 장점 : 코드 간결, 단순 데이터 전달 중심
  - ModelMap
    - 주요 역할 : Model과 거의 동일, Map 기반 데이터 저장
    - 상속 관계 : LinkedHashMap을 상속한 클래스
    - 뷰 지정 방식 : View 이름을 문자열로 리턴
    - 장점 : Map 구조라 직관적, 유연함.
  - ModelAndView
    - 주요 역할 : 데이터(Model) + 뷰(View) 정보를 함께 관리
    - 상속 관계 : 별도의 독립 클래스
    - 뷰 지정 방식 : modelAndView.setViewName()으로 지정.
    - 장점 : 데이터와 뷰를 함께 제어 가능 (복합 제어에 적합)